

**Oracle Database 11g:  
入門 SQL 基礎 I Ed 1**

Volume 2

D49996JP10  
Production 1.0  
November 2007  
D53412

**ORACLE®**

## 原著者

Puja Singh

## 日本語版翻訳監修

郭 峰

© Oracle Corporation 2007 © 日本オラクル株式会社 2007

この文書には、米国Oracle Corporation及び日本オラクル株式会社が権利を有する情報が含まれており、使用と開示に対して定められたライセンス契約に従って提供されるものです。また、これらは著作権法による保護も受けています。ソフトウェアのリバース・エンジニアリングは禁止されています。

この文書が合衆国政府の国防省関連機関に配布される場合は、次の制限付き権利が適用されます。

Use, duplication or disclosure by the Government is subject to restrictions for commercial computer software and shall be deemed to be Restricted Rights software under Federal law, as set forth in subparagraph (c)(1)(ii) of DFARS 252.227-7013, Rights in Technical Data and Computer Software (October 1988).

当社の事前の書面による承諾なしに、いかなる形式あるいはいかなる方法でも、本書及び本書に付属する資料の全体又は一部を複製することを禁じます。いかなる複製も著作権法違反であり、民事または刑事、もしくは両者の制裁の対象となります。

この文書が合衆国政府の国防総省以外の機関に配布される場合は、その権利は、FAR 52.227-14「一般データにおける権利」(Alternate IIIを含む)(June 1987)で定める権利の制限を受けます。

この文書の内容は予告なく変更されることがあります。

米国Oracle Corporation及び日本オラクル株式会社は、本書及び本書に付属する資料についてその記載内容に誤りがない事及び特定目的に対する適合性に関するいっさいの保証を行うものではありません。また、本書を参考にアプリケーション・ソフトウェアを作成された場合であっても、そのアプリケーション・ソフトウェアに関して米国Oracle Corporation及び日本オラクル株式会社(その関連会社も含みます)は一切の責任を負いかねます。

ORACLEは、Oracle Corporationおよびその関連会社の登録商標です。

本書で参照されているその他全ての製品やサービスの名称は、それぞれを表示する為に引用されており、それぞれ各社の商標である場合があります。

Oracle Internal & Oracle Academy  
Use Only

# 目次

<b>I はじめに</b>	
章の目的	I- 2
章の講義項目	I- 3
コースの目的	I- 4
コースの講義項目	I- 5
このコースで使用する付録	I- 7
章の講義項目	I- 8
Oracle Database 11 <sup>g</sup> 対象分野	I- 9
Oracle Database 11 <sup>g</sup>	I-10
Oracle Fusion Middleware	I-12
Oracle Enterprise Manager Grid Control 10 <sup>g</sup>	I-13
Oracle BI Publisher	I-14
章の講義項目	I-15
リレーショナルおよびオブジェクト・リレーショナル・データベース管理システム	I-16
異なるメディアへのデータの格納	I-17
リレーショナル・データベースの概念	I-18
リレーショナル・データベースの定義	I-19
データ・モデル	I-20
E-Rモデル	I-21
エンティティ・リレーションシップモデリング規則	I-23
複数の表の関係付け	I-25
リレーショナル・データベースの用語	I-27
章の講義項目	I-29
SQLを使用したデータベースの問合せ	I-30
SQL文	I-31
SQLの開発環境	I-32
章の講義項目	I-33
Oracle SQL Developerの概要	I-34
Oracle SQL Developerの仕様	I-35
Oracle SQL Developerのインタフェース	I-36
データベース接続の作成	I-38
データベース・オブジェクトの参照	I-41
SQL Worksheetの使用方法	I-42
SQL文の実行	I-45
SQLコードの書式設定	I-46
SQL文の保存	I-47
スクリプト・ファイルの実行	I-48
Oracle SQL DeveloperからのSQL*Plusの起動	I-49

SQL*Plus内のSQL文	I-50
章の講義項目	I-51
Human Resources (HR)スキーマ	I-52
このコースで使用する表	I-53
章の講義項目	I-54
Oracle Database 11gのドキュメント	I-55
その他のリソース	I-56
まとめ	I-57
演習I: 概要	I-58
<b>1 SQL SELECT文を使用したデータの検索</b>	
目的	1- 2
章の講義項目	1- 3
SQL SELECT文の機能	1- 4
基本的なSELECT文	1- 5
すべての列の選択	1- 6
特定の列の選択	1- 7
SQL文の記述	1- 8
列ヘッダーのデフォルト設定	1- 9
章の講義項目	1-10
算術式	1-11
算術演算子の使用方法	1-12
演算子の優先順位	1-13
NULL値の定義	1-14
算術式のNULL値	1-15
章の講義項目	1-16
列別名の定義	1-17
列別名の使用方法	1-18
章の講義項目	1-19
連結演算子	1-20
リテラル文字列	1-21
リテラル文字列の使用方法	1-22
代替引用符(q)演算子	1-23
重複行	1-24
章の講義項目	1-25
表構造の表示	1-26
DESCRIBEコマンドの使用方法	1-27
まとめ	1-28
演習1: 概要	1-29

## 2 データの制限とソート

目的	2- 2
章の講義項目	2- 3
選択を使用した行の制限	2- 4
選択される行の制限	2- 5
WHERE句の使用方法	2- 6
文字列と日付	2- 7
比較演算子	2- 8
比較演算子の使用方法	2- 9
BETWEEN演算子を使用した範囲条件	2-10
IN演算子を使用したメンバーシップ条件	2-11
LIKE演算子を使用したパターン一致	2-12
ワイルドカード文字の組合せ	2-13
NULL条件の使用方法	2-14
論理演算子を使用した条件の定義	2-15
AND演算子の使用方法	2-16
OR演算子の使用方法	2-17
NOT演算子の使用方法	2-18
章の講義項目	2-19
優先順位の規則	2-20
章の講義項目	2-22
ORDER BY句の使用方法	2-23
ソート	2-24
章の講義項目	2-26
置換変数	2-27
シングルアンパサンド置換変数の使用方法	2-29
置換変数での文字値および日付値の使用	2-31
列名、式およびテキストの指定	2-32
ダブルアンパサンド置換変数の使用方法	2-33
章の講義項目	2-34
DEFINEコマンドの使用方法	2-35
VERIFYコマンドの使用方法	2-36
まとめ	2-37
演習2: 概要	2-38

## 3 単一行関数を使用した出力のカスタマイズ

目的	3- 2
章の講義項目	3- 3
SQL関数	3- 4
SQL関数の2つのタイプ	3- 5

単一行関数	3- 6
章の講義項目	3- 8
文字関数	3- 9
大文字/小文字変換関数	3-11
大文字/小文字変換関数の使用方法	3-12
文字操作関数	3-13
文字操作関数の使用方法	3-14
章の講義項目	3-15
数値関数	3-16
ROUND関数の使用方法	3-17
TRUNC関数の使用方法	3-18
MOD関数の使用方法	3-19
章の講義項目	3-20
日付の操作	3-21
RR日付書式	3-22
SYSDATE関数の使用方法	3-24
日付の算術演算	3-25
日付を使用した算術演算子の使用方法	3-26
章の講義項目	3-27
日付操作関数	3-28
日付関数の使用方法	3-29
日付を使用したROUND関数およびTRUNC関数の使用方法	3-30
まとめ	3-31
演習3: 概要	3-32
<b>4 変換関数と条件式の使用方法</b>	
目的	4- 2
章の講義項目	4- 3
変換関数	4- 4
暗黙的なデータ型変換	4- 5
明示的なデータ型変換	4- 7
章の講義項目	4-10
日付を使用したTO_CHAR関数の使用方法	4-11
日付書式モデルの要素	4-12
日付を使用したTO_CHAR関数の使用方法	4-16
数値を使用したTO_CHAR関数の使用方法	4-17
TO_NUMBERおよびTO_DATE関数の使用方法	4-20
RR日付書式を使用したTO_CHARおよびTO_DATE関数の使用方法	4-22
章の講義項目	4-23
関数のネスト	4-24

章の講義項目	4-26
汎用関数	4-27
NVL関数	4-28
NVL関数の使用方法	4-29
NVL2関数の使用方法	4-30
NULLIF関数の使用方法	4-31
COALESCE関数の使用方法	4-32
章の講義項目	4-35
条件式	4-36
CASE式	4-37
CASE式の使用法	4-38
DECODE関数	4-39
DECODE関数の使用方法	4-40
まとめ	4-42
演習4: 概要	4-43
<b>5 グループ関数を使用した集計データのレポート</b>	
目的	5- 2
章の講義項目	5- 3
グループ関数の概要	5- 4
グループ関数のタイプ	5- 5
グループ関数: 構文	5- 6
AVGおよびSUM関数の使用方法	5- 7
MINおよびMAX関数の使用方法	5- 8
COUNT関数の使用方法	5- 9
DISTINCTキーワードの使用法	5-10
グループ関数とNULL値	5-11
章の講義項目	5-12
データのグループの作成	5-13
データのグループの作成: GROUP BY句の構文	5-14
GROUP BY句の使用法	5-15
複数の列によるグループ化	5-17
複数の列に対するGROUP BY句の使用法	5-18
グループ関数を使用した無効な問合せ	5-19
グループの結果の制限	5-21
HAVING句を使用したグループの結果の制限	5-22
HAVING句の使用法	5-23
章の講義項目	5-25
グループ関数のネスト	5-26

まとめ	5-27
演習5: 概要	5-28
<b>6 複数の表のデータの表示</b>	
目的	6- 2
章の講義項目	6- 3
複数の表からのデータの取得	6- 4
結合のタイプ	6- 5
SQL:1999構文を使用した表の結合	6- 6
あいまいな列名の修飾	6- 7
章の講義項目	6- 8
自然結合の作成	6- 9
自然結合によるレコードの取得	6-10
USING句による結合の作成	6-11
列名の結合	6-12
USING句によるレコードの取得	6-13
USING句での表別名の使用方法	6-14
ON句による結合の作成	6-15
ON句によるレコードの取得	6-16
ON句による3方向結合の作成	6-17
結合への追加条件の適用	6-18
章の講義項目	6-19
表自体の結合	6-20
ON句による自己結合の作成	6-21
章の講義項目	6-22
非等価結合	6-23
非等価結合によるレコードの取得	6-24
章の講義項目	6-25
外部結合による直接一致しないレコードの取得	6-26
内部結合と外部結合	6-27
左側外部結合 (LEFT OUTER JOIN)	6-28
右側外部結合 (RIGHT OUTER JOIN)	6-29
完全外部結合 (FULL OUTER JOIN)	6-30
章の講義項目	6-31
デカルト積	6-32
デカルト積の生成	6-33
クロス結合の作成	6-34
まとめ	6-35
演習6: 概要	6-36



<b>7 副問合せによる問合せの解決方法</b>	
目的	7- 2
章の講義項目	7- 3
副問合せによる問題の解決方法	7- 4
副問合せの構文	7- 5
副問合せの使用方法	7- 6
副問合せの使用に関するガイドライン	7- 7
副問合せの種類	7- 8
章の講義項目	7- 9
単一行副問合せ	7-10
単一行副問合せの実行	7-11
副問合せにおけるグループ関数の使用方法	7-12
HAVING句での副問合せ	7-13
この文の間違ひは何か	7-14
内側の問合せから行が戻されない	7-15
章の講義項目	7-16
複数行副問合せ	7-17
複数行副問合せでのANY演算子の使用方法	7-18
複数行副問合せでのALL演算子の使用方法	7-19
章の講義項目	7-20
副問合せにおけるNULL値	7-21
まとめ	7-23
演習7: 概要	7-24
<b>8 集合演算子の使用方法</b>	
目的	8- 2
章の講義項目	8- 3
集合演算子	8- 4
集合演算子のガイドライン	8- 5
Oracleサーバーと集合演算子	8- 6
章の講義項目	8- 7
この章で使用する表	8- 8
章の講義項目	8-12
UNION演算子	8-13
UNION演算子の使用方法	8-14
UNION ALL演算子	8-16
UNION ALL演算子の使用方法	8-17
章の講義項目	8-18
INTERSECT演算子	8-19
INTERSECT演算子の使用方法	8-20

章の講義項目	8-21
MINUS演算子	8-22
MINUS演算子の使用方法	8-23
章の講義項目	8-24
SELECT文の一致	8-25
SELECT文の一致: 例	8-26
章の講義項目	8-27
集合演算子でのORDER BY句の使用方法	8-28
まとめ	8-29
演習8: 概要	8-30
<b>9 データの操作</b>	
目的	9- 2
章の講義項目	9- 3
データ操作言語	9- 4
表への新しい行の追加	9- 5
INSERT文の構文	9- 6
新しい行の挿入	9- 7
NULL値を持つ行の挿入	9- 8
特殊な値の挿入	9- 9
特定の日付値および時刻値の挿入	9-10
スクリプトの作成	9-11
別の表からの行のコピー	9-12
章の講義項目	9-13
表内のデータの変更	9-14
UPDATE文の構文	9-15
表内の行の更新	9-16
副問合せによる2列の更新	9-17
別の表に基づく行の更新	9-18
章の講義項目	9-19
表からの行の削除	9-20
DELETE文	9-21
表からの行の削除	9-22
別の表に基づく行の削除	9-23
TRUNCATE文	9-24
章の講義項目	9-25
データベース・トランザクション	9-26
データベース・トランザクション: 開始と終了	9-27
COMMIT文およびROLLBACK文の利点	9-28
明示的なトランザクション制御文	9-29
マーカーへの変更のロールバック	9-30
暗黙的なトランザクション処理	9-31
COMMITまたはROLLBACK前のデータの状態	9-33
COMMIT後のデータの状態	9-34

データのコミット	9-35
ROLLBACK後のデータの状態	9-36
ROLLBACK後のデータの状態: 例	9-37
文レベル・ロールバック	9-38
章の講義項目	9-39
読取り一貫性	9-40
読取り一貫性の実装	9-41
章の講義項目	9-42
SELECT文のFOR UPDATE句	9-43
FOR UPDATE句: 例	9-44
まとめ	9-46
演習9: 概要	9-47
<b>10 DDL文を使用した表の作成および管理</b>	
目的	10- 2
章の講義項目	10- 3
データベース・オブジェクト	10- 4
ネーミング規則	10- 5
章の講義項目	10- 6
CREATE TABLE文	10- 7
別のユーザーの表の参照	10- 8
DEFAULTオプション	10- 9
表の作成	10-10
章の講義項目	10-11
データ型	10-12
日時データ型	10-14
章の講義項目	10-15
制約の設定	10-16
制約のガイドライン	10-17
制約の定義	10-18
NOT NULL制約	10-20
UNIQUE制約	10-21
PRIMARY KEY制約	10-23
FOREIGN KEY制約	10-24
FOREIGN KEY制約: キーワード	10-26
CHECK制約	10-27
CREATE TABLE: 例	10-28
制約違反	10-29
章の講義項目	10-31
副問合せを使用した表の作成	10-32

章の講義項目	10-34
ALTER TABLE文	10-35
読取り専用の表	10-36
章の講義項目	10-37
表の削除	10-38
まとめ	10-39
演習10: 概要	10-40
<b>11 その他のスキーマ・オブジェクトの作成</b>	
目的	11- 2
章の講義項目	11- 3
データベース・オブジェクト	11- 4
ビューの概要	11- 5
ビューの利点	11- 6
単一ビューと複合ビュー	11- 7
ビューの作成	11- 8
ビューからのデータの取得	11-11
ビューの変更	11-12
複合ビューの作成	11-13
ビューでDML操作を実行するためのルール	11-14
WITH CHECK OPTION句の使用方法	11-17
DML操作の拒否	11-18
ビューの削除	11-20
演習11: パート1の概要	11-21
章の講義項目	11-22
順序	11-23
CREATE SEQUENCE文: 構文	11-25
順序の作成	11-26
NEXTVALおよびCURRVAL疑似列	11-27
順序の使用方法	11-29
順序値のキャッシュ	11-30
順序の変更	11-31
順序変更のガイドライン	11-32
章の講義項目	11-33
索引	11-34
索引の作成方法	11-36
索引の作成	11-37
索引作成のガイドライン	11-38
索引の削除	11-39
章の講義項目	11-40

シノニム	11-41
オブジェクトのシノニムの作成	11-42
シノニムの作成および削除	11-43
まとめ	11-44
演習11: パート2の概要	11-45

**付録A: 演習の解答**

**付録B: 表の説明**

**付録C: Oracle結合構文**

**付録D: SQL\*Plusの使用**

**付録E: SQL Developer GUIを使用したDMLおよびDDL操作の実行**

**追加の演習**

**追加の演習: 解答**

Oracle Internal & Oracle Academy  
Use Only

Oracle Internal & Oracle Academy  
Use Only

# データの操作

ORACLE

Copyright © 2007, Oracle. All rights reserved.

Oracle Internal & Oracle Academy  
Use Only

## 目的

この章を終えると、次のことができるようになります。

- 各データ操作言語 (DML) 文の説明
- 表への行の挿入
- 表内の行の更新
- 表からの行の削除
- トランザクションの制御

ORACLE

Copyright © 2007, Oracle. All rights reserved.

### 目的

この章では、データ操作言語 (DML) 文による表への行の挿入、表内の既存行の更新、および表からの既存行の削除の方法について学習します。また、COMMIT 文、SAVEPOINT 文および ROLLBACK 文によるトランザクションの制御方法についても学習します。



## 章の講義項目

- 表への新しい行の追加
  - INSERT文
- 表内のデータの変更
  - UPDATE文
- 表からの行の削除
  - DELETE文
  - TRUNCATE文
- COMMIT、ROLLBACKおよびSAVEPOINTによるデータベース・トランザクション制御
- 読取り一貫性
- SELECT文のFOR UPDATE句

ORACLE

Copyright © 2007, Oracle. All rights reserved.

# データ操作言語

- 次の操作を行う場合にDML文を実行します。
  - 表に新しい行を追加する
  - 表の既存の行を変更する
  - 表から既存の行を削除する
- トランザクションは、作業の論理単位を形成するDML文のコレクションで構成されます。

ORACLE

Copyright © 2007, Oracle. All rights reserved.

## データ操作言語

データ操作言語 (DML) は、SQLのコア部分です。データベースでのデータの追加、更新または削除を行う場合には、DML文を実行します。作業の論理単位を形成するDML文のコレクションをトランザクションと呼びます。

銀行のデータベースで考えてみます。銀行の顧客が普通預金から当座預金に資金を移動する場合、トランザクションは、普通預金の減額、当座預金の増額、およびトランザクション・ジャーナルへのこのトランザクションの記録という3つの独立した処理で構成されることになります。Oracleサーバーでは、この3つのSQL文がすべて実行されて、これらの口座の差引残高が正しく維持されることを保証する必要があります。なんらかの原因でトランザクションの文のいずれかが実行されない場合には、トランザクションのその他の文が元に戻されるようにする必要があります。

**注意:** この章のほとんどのDML文は、表に関する制約に違反していないことを前提としています。制約については、このコースの後の方で説明します。

**注意:** SQL Developerで、「Run Script」アイコンをクリックするか、[F5]を押してDML文を実行します。「Script Output」タブ・ページにフィードバック・メッセージが表示されます。

## 表への新しい行の追加

70 Public Relations	100	1700
---------------------	-----	------

新しい行

# DEPARTMENTS

DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
1	10 Administration	200	1700
2	20 Marketing	201	1800
3	50 Shipping	124	1500
4	60 IT	103	1400
5	80 Sales	149	2500
6	90 Executive	100	1700
7	110 Accounting	205	1700
8	190 Contracting	(null)	1700

新しい行を  
DEPARTMENTS表に  
挿入

DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
1	10 Administration	200	1700
2	20 Marketing	201	1800
3	50 Shipping	124	1500
4	60 IT	103	1400
5	80 Sales	149	2500
6	90 Executive	100	1700
7	110 Accounting	205	1700
8	190 Contracting	(null)	1700

9	70 Public Relations	100	1700
---	---------------------	-----	------

Copyright © 2007, Oracle. All rights reserved.

### 表への新しい行の追加

スライドの図では、新しい部門がDEPARTMENTS表に追加されています。

## INSERT文の構文

- 表に新しい行を追加するには、INSERT文を使用します。

```
INSERT INTO  table [(column [, column...])]
VALUES      (value [, value...]);
```

- この構文の場合、一度に1行のみ追加されます。

ORACLE

Copyright © 2007, Oracle. All rights reserved.

### INSERT文の構文

INSERT文を発行すると、新しい行を表に追加できます。

構文の内容

<i>table</i>	表の名前
<i>column</i>	移入先の表の列の名前
<i>value</i>	列に対応する値

**注意:** VALUES句を含む文では、一度に1行のみ表に追加されます。

## 新しい行の挿入

- 各列の値を含む新しい行を挿入します。
- 表内の列のデフォルトの順序で、値のリストを指定します。
- オプションで、INSERT句に列のリストを指定します。

```
INSERT INTO departments(department_id,  
                        department_name, manager_id, location_id)  
VALUES (70, 'Public Relations', 100, 1700);  
1 rows inserted
```

- 文字値および日付値は一重引用符で囲みます。

ORACLE

Copyright © 2007, Oracle. All rights reserved.

### 新しい行の挿入

INSERT句では各列の値を含む新しい行を挿入できるため、列リストは必須ではありません。ただし、列リストを使用しない場合は、表内の列のデフォルトの順序に従って値のリストを指定し、各列に値を割り当てる必要があります。

DESCRIBE departments

Name	Null	Type
DEPARTMENT_ID	NOT NULL	NUMBER(4)
DEPARTMENT_NAME	NOT NULL	VARCHAR2(30)
MANAGER_ID		NUMBER(6)
LOCATION_ID		NUMBER(4)

明確にするには、INSERT句で列リストを使用します。

文字値および日付値は一重引用符で囲みます。ただし、数値を一重引用符で囲むことはお薦めできません。

## NULL値を持つ行の挿入

- 暗黙的手法: 列リストから列を省略する。

```
INSERT INTO departments (department_id,  
                           department_name)  
VALUES (30, 'Purchasing');  
1 rows inserted
```

- 明示的手法: VALUES句にNULLキーワードを指定する。

```
INSERT INTO departments  
VALUES (100, 'Finance', NULL, NULL);  
1 rows inserted
```

ORACLE

Copyright © 2007, Oracle. All rights reserved.

### NULL値を持つ行の挿入

手法	説明
暗黙的手法	列リストから列を省略します。
明示的手法	NULLキーワードをVALUESリストに指定します。 文字列および日付に対しては、VALUESリストで空文字列('')を指定します。

DESCRIBEコマンドでNULLステータスを確認して、目的の列でNULL値を使用できるかどうかを確認してください。

Oracleサーバーでは、データ型、データ範囲およびデータ整合性のすべての制約が自動的に適用されます。明示的にリストに指定されていない列は、新しい行にNULL値が割り当てられます。ユーザー入力で発生する可能性のある一般的なエラーは、次の順序でチェックされます。

- NOT NULL列での必須の値の欠落
- 重複値による一意キー制約または主キー制約の違反
- Any値によるCHECK制約の違反
- 外部キー制約のための参照整合性の維持
- データ型の不一致または列に収まらない長すぎる値

**注意:** INSERT文を分かりやすく、信頼性を向上させるため、または間違いの発生を低減するためにも、列リストの使用をお勧めします。

## 特殊な値の挿入

SYSDATE関数では、現在の日付と時刻が記録されます。

```
INSERT INTO employees (employee_id,
                        first_name, last_name,
                        email, phone_number,
                        hire_date, job_id, salary,
                        commission_pct, manager_id,
                        department_id)
VALUES (113,
        'Louis', 'Popp',
        'LPOPP', '515.124.4567',
        SYSDATE, 'AC_ACCOUNT', 6900,
        NULL, 205, 110);
```

1 rows inserted

ORACLE

Copyright © 2007, Oracle. All rights reserved.

### 特殊な値の挿入

関数を使用して、特殊な値を表に入力できます。

スライドの例では、従業員Poppの情報がEMPLOYEES表に記録されます。この処理では、現在の日付と時刻がHIRE\_DATE列に提供されます。これには、データベース・サーバーの現在の日付と時刻を戻すSYSDATE関数を使用されています。また、セッション・タイムゾーンの現在の日付を取得する場合は、CURRENT\_DATE関数を利用できます。行を表に挿入するときにUSER関数も使用できます。USER関数は、現行のユーザー名を記録します。

#### 表への追加結果の確認

```
SELECT employee_id, last_name, job_id, hire_date, commission_pct
FROM   employees
WHERE  employee_id = 113;
```

	EMPLOYEE_ID	LAST_NAME	JOB_ID	HIRE_DATE	COMMISSION_PCT
1	113	Popp	AC_ACCOUNT	11-JUN-07	(null)

## 特定の日付値および時刻値の挿入

- 新しい従業員を追加します。

```
INSERT INTO employees
VALUES      (114,
             'Den', 'Raphealy',
             'DRAPHEAL', '515.127.4561',
             TO_DATE('FEB 3, 1999', 'MON DD, YYYY'),
             'SA_REP', 11000, 0.2, 100, 60);
```

1 rows inserted

- 追加結果を確認します。

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	COMMISSION_PCT
114	Den	Raphealy	DRAPHEAL	515.127.4561	03-FEB-99	SA_REP	11000	0.2

ORACLE

Copyright © 2007, Oracle. All rights reserved.

### 特定の日付値および時刻値の挿入

通常、日付値の挿入にはDD-MON-RR書式が使用されます。RR書式の場合、正しい西暦が自動的にシステムから提供されます。

日付値をDD-MON-YYYY書式で提供することもできます。この場合、西暦が明確に指定され、正しい西暦を指定するための内部的なRR書式のロジックに依存しないため、この書式の使用をお勧めします。

デフォルト書式以外の書式で日付を入力する必要がある場合（たとえば、別の西暦または特定の時刻を入力する場合）、TO\_DATE関数を使用する必要があります。

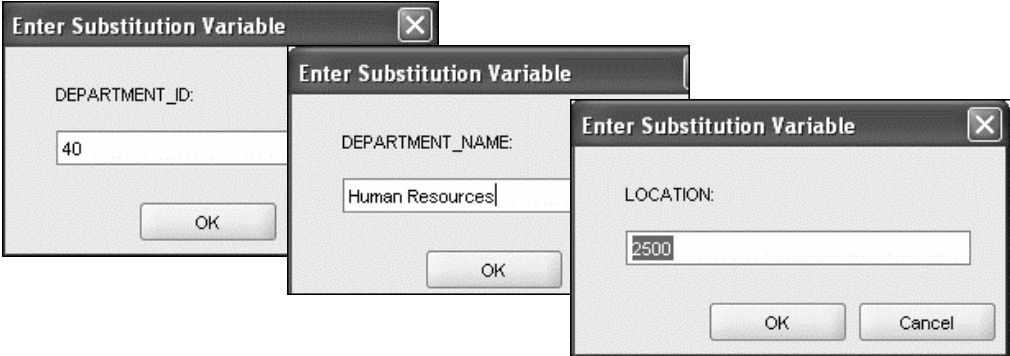
スライドの例では、従業員Raphealyの情報がEMPLOYEES表に記録され、HIRE\_DATE列には「February 3, 1999」が設定されます。



## スクリプトの作成

- 値の入力を求めるには、SQL文で&置換を使用します。
- &は変数値のプレースホルダです。

```
INSERT INTO departments
      (department_id, department_name, location_id)
VALUES (&department_id, '&department_name', &location);
```



The image shows three overlapping 'Enter Substitution Variable' dialog boxes. The first dialog is for 'DEPARTMENT\_ID' with the value '40'. The second dialog is for 'DEPARTMENT\_NAME' with the value 'Human Resources'. The third dialog is for 'LOCATION' with the value '2500'.

ORACLE

Copyright © 2007, Oracle. All rights reserved.

### スクリプトの作成

置換変数を使用したコマンドをファイルに保存して、そのファイル内のコマンドを実行することができます。スライドの例では、部門の情報がDEPARTMENTS表に記録されます。

スクリプト・ファイルを実行すると、アンパサンド(&)置換変数のそれぞれに対して入力を求めるプロンプトが表示されます。置換変数に値を入力した後、「OK」ボタンをクリックします。入力した値は、文に代入されます。これにより、同じスクリプト・ファイルを繰り返し実行し、その都度、一連の別の値を指定することが可能になります。

## 別の表からの行のコピー

- INSERT文に副問合せを記述します。

```
INSERT INTO sales_reps(id, name, salary, commission_pct)
  (SELECT employee_id, last_name, salary, commission_pct
   FROM   employees
   WHERE  job_id LIKE '%REP%');
```

4 rows inserted

- VALUES句は使用しません。
- INSERT句の列数と副問合せの列数を一致させます。
- 副問合せによって戻される行をすべてsales\_reps表に挿入します。

ORACLE

Copyright © 2007, Oracle. All rights reserved.

### 別の表からの行のコピー

INSERT文で値を既存の表から取得して、行を表に追加することができます。スライドの例のINSERT INTO文が動作するには、CREATE TABLE文を使用してあらかじめsales\_reps表を作成しておく必要があります。CREATE TABLE文については、DDL文を使用した表の作成および管理に関する次の章で説明します。

VALUES句の代わりに、副問合せを使用します。

#### 構文

```
INSERT INTO table [ column (, column) ] subquery;
```

#### 構文の内容

<i>table</i>	表の名前
<i>column</i>	移入先の表の列の名前
<i>subquery</i>	行を表に戻す副問合せ

INSERT句の列リストの列の数とデータ型は、副問合せの値の数とデータ型に一致する必要があります。副問合せによって戻される行数に応じて、0行以上の行が追加されます。表の行のコピーを作成するには、副問合せで次のようにSELECT \*を使用します。

```
INSERT INTO copy_emp
  SELECT *
  FROM   employees;
```

## 章の講義項目

- 表への新しい行の追加
  - INSERT文
- 表内のデータの変更
  - UPDATE文
- 表からの行の削除
  - DELETE文
  - TRUNCATE文
- COMMIT、ROLLBACKおよびSAVEPOINTによるデータベース・トランザクション制御
- 読取り一貫性
- SELECT文のFOR UPDATE句

ORACLE

Copyright © 2007, Oracle. All rights reserved.

## 表内のデータの変更

### EMPLOYEES

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	SALARY	MANAGER_ID	COMMISSION_PCT	DEPARTMENT_ID
100	Steven	King	24000	(null)	(null)	90
101	Neena	Kochhar	17000	100	(null)	90
102	Lex	De Haan	17000	100	(null)	90
103	Alexander	Hunold	9000	102	(null)	60
104	Bruce	Ernst	6000	103	(null)	60
107	Diana	Lorentz	4200	103	(null)	60
124	Kevin	Mourgos	5800	100	(null)	50

次のように、EMPLOYEES表の行を更新します。 →

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	SALARY	MANAGER_ID	COMMISSION_PCT	DEPARTMENT_ID
100	Steven	King	24000	(null)	(null)	90
101	Neena	Kochhar	17000	100	(null)	90
102	Lex	De Haan	17000	100	(null)	90
103	Alexander	Hunold	9000	102	(null)	80
104	Bruce	Ernst	6000	103	(null)	80
107	Diana	Lorentz	4200	103	(null)	80
124	Kevin	Mourgos	5800	100	(null)	50

ORACLE

Copyright © 2007, Oracle. All rights reserved.

### 表内のデータの変更

スライドでは、部門60の従業員の部門番号が部門80に変更されます。

## UPDATE文の構文

- 表内の既存の値を変更するには、UPDATE文を使用します。

UPDATE	<i>table</i>
SET	<i>column = value [, column = value, ...]</i>
[WHERE	<i>condition</i> ];

- 複数の行が一度に更新されます(必要に応じて)。

ORACLE

Copyright © 2007, Oracle. All rights reserved.

### UPDATE文の構文

UPDATE文を使用すると、表内の既存の値を変更できます。

構文の内容

<i>table</i>	表の名前
<i>column</i>	移入先の表の列の名前
<i>value</i>	列に対応する値または副問合せ
<i>condition</i>	更新する行を特定します。列名、式、定数、副問合せおよび比較演算子で指定します。

更新操作の結果を確認するには、表を問合せで更新した行を表示します。

詳細は、『Oracle Database SQL言語リファレンス11gリリース1(11.1)』のUPDATEに関する節を参照してください。

**注意:** 一般に、更新する単一行を特定する場合は、WHERE句で主キー列を使用します。それ以外の列を使用すると、予想外に複数の行が更新される可能性があります。たとえば、EMPLOYEES表の単一行を名前で特定するのは危険です。同じ名前の従業員が複数存在する可能性があるためです。

## 表内の行の更新

- WHERE句を指定すると、特定の1行または複数行の値が変更されます。

```
UPDATE employees
SET    department_id = 50
WHERE  employee_id = 113;
```

1 rows updated

- WHERE句を省略すると、表内のすべての行の値が変更されます。

```
UPDATE    copy_emp
SET       department_id = 110;
```

22 rows updated

- 列の値をNULLに更新するには、SET *column\_name* = NULLを指定します。

ORACLE

Copyright © 2007, Oracle. All rights reserved.

### 表内の行の更新

UPDATE文でWHERE句を指定した場合、特定の1行または複数行の値が変更されます。スライドの例では、従業員113(Popp)が部門50に変更されます。

WHERE句を省略すると、表内のすべての行の値が変更されます。COPY\_EMP表の行で更新結果を確認してください。

```
SELECT last_name, department_id
FROM    copy_emp;
```

	LAST_NAME	DEPARTMENT_ID
1	King	110
2	Kochhar	110

...

たとえば、これまでSA\_REPであった従業員の職務がIT\_PROGに変更されているとします。そのため、この従業員のJOB\_IDを更新し、歩合フィールドをNULLに設定する必要があります。

```
UPDATE employees
SET job_id = 'IT_PROG', commission_pct = NULL
WHERE employee_id = 114;
```

注意: COPY\_EMP表のデータは、EMPLOYEES表のデータと同一です。

## 副問合せによる2列の更新

従業員113の職務と給与を更新して、従業員205の職務と給与に合わせます。

```
UPDATE employees
SET   job_id = (SELECT job_id
                FROM   employees
                WHERE  employee_id = 205),
      salary = (SELECT salary
                FROM   employees
                WHERE  employee_id = 205)
WHERE employee_id = 113;

1 rows updated
```

ORACLE

Copyright © 2007, Oracle. All rights reserved.

### 副問合せによる2列の更新

UPDATE文のSET句に複数の副問合せを記述すると、複数列の更新ができます。構文は次のとおりです。

```
UPDATE table
SET   column =
      (SELECT column
       FROM table
       WHERE condition)
[ ,
  column =
      (SELECT column
       FROM table
       WHERE condition) ]
[WHERE condition] ;
```

スライドの例は、次のように記述することもできます。

```
UPDATE employees
SET (job_id, salary) = (SELECT job_id, salary
                       FROM   employees
                       WHERE  employee_id = 205)
WHERE employee_id = 113;
```

## 別の表に基づく行の更新

別の表の値に基づいて表の行の値を更新するには、UPDATE文で副問合せを使用します。

```
UPDATE copy_emp
SET    department_id = (SELECT department_id
                        FROM employees
                        WHERE employee_id = 100)
WHERE  job_id        = (SELECT job_id
                        FROM employees
                        WHERE employee_id = 200);

1 rows updated
```

ORACLE

Copyright © 2007, Oracle. All rights reserved.

### 別の表に基づく行の更新

UPDATE文で副問合せを使用すると、表の値を更新できます。スライドの例では、EMPLOYEES表の値に基づいてCOPY\_EMP表が更新されます。従業員200の職務IDを持つすべての従業員の部門番号が、従業員100の現行の部門番号に変更されます。



## 章の講義項目

- 表への新しい行の追加
  - INSERT文
- 表内のデータの変更
  - UPDATE文
- 表からの行の削除
  - DELETE文
  - TRUNCATE文
- COMMIT、ROLLBACKおよびSAVEPOINTによるデータベース・トランザクション制御
- 読取り一貫性
- SELECT文のFOR UPDATE句

ORACLE

Copyright © 2007, Oracle. All rights reserved.

## 表からの行の削除

### DEPARTMENTS

DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
1	10 Administration	200	1700
2	20 Marketing	201	1800
3	50 Shipping	124	1500
4	60 IT	103	1400
5	80 Sales	149	2500
6	90 Executive	100	1700
7	110 Accounting	205	1700
8	190 Contracting	(null)	1700

次のように、DEPARTMENTS表から行を削除します。

DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
1	10 Administration	200	1700
2	20 Marketing	201	1800
3	50 Shipping	124	1500
4	60 IT	103	1400
5	80 Sales	149	2500
6	90 Executive	100	1700
7	110 Accounting	205	1700

ORACLE

Copyright © 2007, Oracle. All rights reserved.

### 表からの行の削除

スライドの図に示すように、Contracting部門がDEPARTMENTS表から削除されました。DEPARTMENTS表における制約には違反していないことを前提としています。

# DELETE文

DELETE文を使用すると、既存の行を表から削除できます。

```
DELETE [FROM] table
[WHERE condition];
```

ORACLE

Copyright © 2007, Oracle. All rights reserved.

## DELETE文の構文

DELETE文を使用すると、既存の行を表から削除できます。

構文の内容

*table*                      表の名前

*condition*                削除する行を特定します。列名、式、定数、副問合せおよび比較演算子で指定します。

**注意:** 行が削除されない場合、メッセージ“0 rows deleted”が戻されます (SQL Developerの「Script Output」タブ)。

詳細は、『Oracle Database SQL言語リファレンス11gリリース1 (11.1)』のDELETEに関する節を参照してください。

## 表からの行の削除

- WHERE句を指定すると、特定の行が削除されます。

```
DELETE FROM departments
WHERE department_name = 'Finance';
```

1 rows deleted

- WHERE句を省略すると、表内のすべての行が削除されます。

```
DELETE FROM copy_emp;
```

22 rows deleted

ORACLE

Copyright © 2007, Oracle. All rights reserved.

### 表からの行の削除

DELETE文でWHERE句を指定すると、特定の行を削除できます。スライドの最初の例では、DEPARTMENTS表から経理部門が削除されます。削除した行をSELECT文で表示することで、削除操作の結果を確認できます。

```
SELECT *
FROM departments
WHERE department_name = 'Finance';
```

0 rows selected

一方、WHERE句を省略した場合は、表内のすべての行が削除されます。スライドの2番目の例では、WHERE句が指定されていないため、COPY\_EMP表からすべての行が削除されます。

#### 例

WHERE句で特定される行を削除します。

```
DELETE FROM employees WHERE employee_id = 114;
```

1 rows deleted

```
DELETE FROM departments WHERE department_id IN (30, 40);
```

2 rows deleted

## 別の表に基づく行の削除

別の表の値に基づいて表から行を削除するには、DELETE文で副問合せを使用します。

```
DELETE FROM employees
WHERE department_id =
    (SELECT department_id
     FROM departments
     WHERE department_name
           LIKE '%Public%');
```

1 rows deleted

ORACLE

Copyright © 2007, Oracle. All rights reserved.

### 別の表に基づく行の削除

副問合せを使用すると、別の表の値に基づいて表から行を削除できます。スライドの例では、部門名にAdminという文字列が含まれる部門の従業員がすべて削除されます。副問合せは、DEPARTMENTS表で文字列Publicが含まれる部門名を検索して、その部門番号を取得します。次に、複問合せから主問合せにこの部門番号が渡され、主問合せはこの部門番号に基づいてEMPLOYEES表からデータ行を削除します。

# TRUNCATE文

- 表からすべての行が削除され、表は空となり表の構造はそのまま残ります。
- この文は、DML文ではなくデータ定義言語 (DDL) であるため、簡単には元に戻せません。

- 構文

```
TRUNCATE TABLE table_name;
```

- 例

```
TRUNCATE TABLE copy_emp;
```

ORACLE

Copyright © 2007, Oracle. All rights reserved.

## TRUNCATE文

より効率的に表を空にする方法は、TRUNCATE文を使用することです。

TRUNCATE文では、表またはクラスタからすべての行をすばやく削除できます。TRUNCATE文の方がDELETE文より行の削除が速い理由は次のとおりです。

- TRUNCATE文は、データ定義言語 (DDL) 文であるため、ロールバック情報を生成しません。ロールバック情報については、この章で後述します。
- 表の切捨てでは、表の削除トリガーは起動されません。

表が参照整合性制約の親表である場合、その表を切り捨てることはできません。TRUNCATE文を発行する前に、参照整合性制約を無効にする必要があります。制約の無効化については、次の章で説明します。

## 章の講義項目

- 表への新しい行の追加
  - INSERT文
- 表内のデータの変更
  - UPDATE文
- 表からの行の削除
  - DELETE文
  - TRUNCATE文
- COMMIT、ROLLBACKおよびSAVEPOINTによるデータベース・トランザクション制御
- 読取り一貫性
- SELECT文のFOR UPDATE句

ORACLE

Copyright © 2007, Oracle. All rights reserved.

# データベース・トランザクション

データベース・トランザクションは、次のいずれかで構成されます。

- データに対する1つの一貫性のある変更を構成する複数のDML文
- 1つのDDL文
- 1つのデータ制御言語(DCL)文

ORACLE

Copyright © 2007, Oracle. All rights reserved.

## データベース・トランザクション

Oracleサーバーでは、トランザクションに基づいてデータ整合性が保証されます。トランザクションにより、データ変更時のより一層の柔軟性と制御が得られ、ユーザー・プロセスの失敗やシステム障害が発生した場合でもデータ整合性が保証されます。

トランザクションは、データに対する1つの一貫性のある変更を構成する複数のDML文から成ります。たとえば、2つの口座の間での資金の移動には、1つの口座での借方記帳と、もう1つの口座での同一金額の貸方記帳が含まれるはずです。これらの処理は、両方失敗するか両方成功するかのいずれかである必要があります。借方記帳が行われない場合は、貸方記帳がコミットされないようにします。

### トランザクションの種類

種類	説明
データ操作言語 (DML)	Oracleサーバーで単一エンティティまたは作業の論理単位として扱われる、任意の数のDML文で構成されます。
データ定義言語 (DDL)	1つのDDL文のみで構成されます。
データ制御言語 (DCL)	1つのDCL文のみで構成されます。



## データベース・トランザクション: 開始と終了

- 最初のDML SQL文の実行により開始します。
- 次のいずれかのイベントで終了します。
  - COMMIT文またはROLLBACK文が発行される。
  - DDL文またはDCL文が実行される(自動コミット)。
  - ユーザーがSQL DeveloperまたはSQL\*Plusを終了する。
  - システム・クラッシュが発生する。

ORACLE

Copyright © 2007, Oracle. All rights reserved.

### データベース・トランザクション: 開始と終了

データベース・トランザクションの開始と終了はいつでしょうか?

トランザクションは、最初のDML文が検出されたときに開始され、次のいずれかが発生すると終了します。

- COMMIT文またはROLLBACK文の発行
- DDL文(CREATEなど)の発行
- DCL文の発行
- ユーザーによるSQL DeveloperまたはSQL\*Plusの終了
- マシンの障害やシステム・クラッシュの発生

1つのトランザクションが終了すると、次の実行可能なSQL文により自動的に次のトランザクションが開始されます。

DDL文またはDCL文は自動的にコミットされるため、暗黙的にトランザクションが終了します。

## COMMIT文およびROLLBACK文の利点

COMMIT文およびROLLBACK文を使用すると、次の利点があります。

- データ整合性を保証できる
- 変更を永続的に確定する前に、データの変更をプレビューできる
- 論理的に関連する操作をグループ化できる

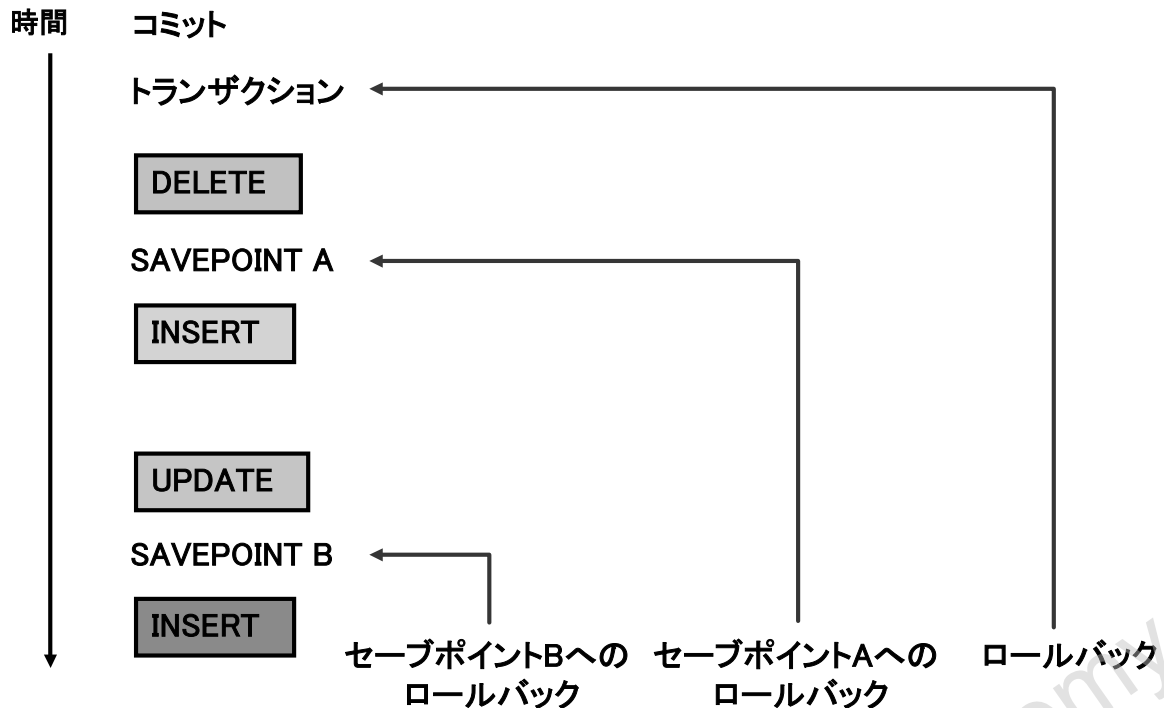
ORACLE

Copyright © 2007, Oracle. All rights reserved.

### COMMIT文およびROLLBACK文の利点

COMMIT文およびROLLBACK文を使用すると、データ変更の確定を制御できます。

## 明示的なトランザクション制御文



ORACLE

Copyright © 2007, Oracle. All rights reserved.

### 明示的なトランザクション制御文

COMMIT文、SAVEPOINT文およびROLLBACK文を使用すると、トランザクションのロジックを制御できます。

文	説明
COMMIT	保留中のデータ変更をすべて永続的に確定し、現行のトランザクションを終了します。
SAVEPOINT <i>name</i>	現行のトランザクション内にセーブポイントのマーカーを設定します。
ROLLBACK	ROLLBACKは、保留中のデータ変更をすべて破棄し、現行のトランザクションを終了します。
ROLLBACK TO <i>SAVEPOINT name</i>	ROLLBACK TO SAVEPOINTは、現行のトランザクションを指定したセーブポイントまでロールバックします。したがって、ロールバック先のセーブポイントより後に作成された変更やセーブポイントはすべて破棄されます。TO SAVEPOINT句を省略した場合、ROLLBACK文はトランザクション全体をロールバックします。セーブポイントは論理的なものであるため、作成したセーブポイントのリストを表示する方法はありません。

**注意:** SAVEPOINTに対してCOMMITを実行することはできません。SAVEPOINTはANSI標準のSQLではありません。

## マーカーへの変更のロールバック

- 現行のトランザクションにマーカーを作成するには、**SAVEPOINT**文を使用します。
- マーカーまでロールバックするには、**ROLLBACK TO SAVEPOINT**文を使用します。

```
UPDATE...  
SAVEPOINT update_done;  
SAVEPOINT update_done succeeded.  
INSERT...  
ROLLBACK TO update_done;  
ROLLBACK TO succeeded.
```

ORACLE

Copyright © 2007, Oracle. All rights reserved.

### マーカーへの変更のロールバック

SAVEPOINT文を使用すると、現行のトランザクションにマーカーを作成できます。マーカーを作成すると、トランザクションが小さなセクションに分割されます。また、ROLLBACK TO SAVEPOINT文を使用して、そのマーカーまでの保留中の変更を破棄することができます。

前に作成したセーブポイントと同じ名前のセーブポイントを新たに作成すると、前のセーブポイントは削除される点に注意してください。

## 暗黙的なトランザクション処理

- 次のような場合に、自動コミットが発生します。
  - DDL文が発行された
  - DCL文が発行された
  - SQL DeveloperまたはSQL\*Plusを正常に終了した  
(COMMIT文またはROLLBACK文の明示的な発行なし)
- SQL DeveloperまたはSQL\*Plusの異常終了やシステム障害が発生した場合は、自動ロールバックが発生します。

ORACLE

Copyright © 2007, Oracle. All rights reserved.

### 暗黙的なトランザクション処理

状態	実行条件
自動コミット	DDL文またはDCL文の発行 SQL DeveloperまたはSQL*Plusの正常終了 (COMMITコマンドまたはROLLBACKコマンドの明示的な発行なし)
自動ロールバック	SQL DeveloperまたはSQL*Plusの異常終了、あるいはシステム障害

**注意:** SQL\*Plusでは、AUTOCOMMITコマンドのONとOFFを切り替えることができます。ONに設定すると、個々のDML文は実行されるとすぐにコミットされます。変更をロールバックすることはできません。OFFに設定すると、COMMIT文を明示的に発行することができます。また、DDL文を発行した場合やSQL\*Plusを終了した場合にも、COMMIT文が発行されます。SET AUTOCOMMIT ON/OFFコマンドはSQL Developerではスキップされます。自動コミット・プリファレンスを有効にしている場合のみ、SQL Developerの正常終了時にDMLがコミットされます。自動コミットを有効にするには、次の操作を実行します。

- 「Tools」メニューで「Preferences」を選択します。「Preferences」ダイアログ・ボックスで「Database」を展開し、「Worksheet Parameters」を選択します。
- 右ペインで「Autocommit in SQL Worksheet」オプションをチェックし、「OK」をクリックします。

## 暗黙的なトランザクション処理(続き)

### システム障害

トランザクションがシステム障害で中断されると、トランザクション全体が自動的にロールバックされます。これにより、エラーが原因による不必要なデータ変更が防止され、最終のコミット時点の状態で表に戻ります。このようにして、Oracleサーバーは表の整合性を保護します。

SQL Developerの場合、「File」メニューから「Exit」を選択するとセッションからの正常終了が実行されます。SQL\*Plusの場合は、プロンプトでEXITコマンドを入力すると正常終了できます。ウィンドウを閉じた場合は、異常終了と解釈されます。

Oracle Internal & Oracle Academy  
Use Only

## COMMITまたはROLLBACK前のデータの状態

- 以前のデータの状態にリカバリ可能です。
- 現行のユーザーは、SELECT文を使用して、DML操作の結果を確認できます。
- 他のユーザーは、現行のユーザーが発行したDML文の結果を見ることはできません。
- 影響を受ける行はロックされます。他のユーザーは影響を受ける行のデータを変更できません。

ORACLE

Copyright © 2007, Oracle. All rights reserved.

### COMMITまたはROLLBACK前のデータの状態

トランザクション中に行われたすべてのデータ変更は、そのトランザクションがコミットされるまで一時的なものです。

COMMIT文またはROLLBACK文が発行される前のデータの状態は、次のように説明できます。

- データ操作は主としてデータベースのバッファに対して実行されます。このため、以前のデータの状態にリカバリすることができます。
- 現行のユーザーは、表を問い合わせることで、データ操作の結果を確認できます。
- 他のユーザーは、現行のユーザーが行ったデータ操作の結果を表示することはできません。Oracleサーバーでは、読取り一貫性を導入して、最後のコミット時点の状態のデータが各ユーザーに表示されることを保証しています。
- 影響を受ける行はロックされます。他のユーザーは影響を受ける行のデータを変更できません。

## COMMIT後のデータの状態

- データの変更内容がデータベースに保存されます。
- 以前のデータの状態に上書きされます。
- すべてのユーザーが結果を表示できます。
- 影響を受ける行のロックが解除されます。他のユーザーはこれらの行を操作可能になります。
- すべてのセーブポイントが消去されます。

ORACLE

Copyright © 2007, Oracle. All rights reserved.

### COMMIT後のデータの状態

COMMIT文を使用することにより、すべての保留中の変更が確定されます。COMMIT文の発行後は次の処理が行われます。

- データの変更内容がデータベースに書き込まれます。
- 通常のSQL問合せでは以前のデータ状態を利用できなくなります。
- すべてのユーザーが結果を表示できます。
- 影響を受ける行のロックが解除されます。他のユーザーは、これらの行に対して新規のデータ変更を実行可能になります。
- すべてのセーブポイントが消去されます。



# データのコミット

- 変更の実施

```
DELETE FROM employees
WHERE employee_id = 99999;
1 rows deleted

INSERT INTO departments
VALUES (290, 'Corporate Tax', NULL, 1700);
1 rows inserted
```

- 変更のコミット

```
COMMIT;
COMMIT succeeded.
```

ORACLE

Copyright © 2007, Oracle. All rights reserved.

## データのコミット

スライドの例では、EMPLOYEES表から1行削除され、DEPARTMENTS表に新しい行が挿入されます。この変更は、COMMIT文を発行することで保存されます。

### 例

DEPARTMENTS表の部門290および300を削除し、EMPLOYEES表の1行を更新します。このデータの変更を保存します。

```
DELETE FROM departments
WHERE department_id IN (290, 300);
```

```
UPDATE employees
SET department_id = 80
WHERE employee_id = 206;
```

```
COMMIT;
```

## ROLLBACK後のデータの状態

ROLLBACK文を使用すると、保留中の変更がすべて破棄されます。

- データの変更内容が元に戻されます。
- 以前のデータの状態にリストアされます。
- 影響を受ける行のロックが解除されます。

```
DELETE FROM copy_emp;  
ROLLBACK ;
```

ORACLE

Copyright © 2007, Oracle. All rights reserved.

### ROLLBACK後のデータの状態

ROLLBACK文を使用すると、保留中の変更がすべて破棄され、結果として次の処理が行われます。

- データの変更内容が元に戻されます。
- 以前のデータの状態にリストアされます。
- 影響を受ける行のロックが解除されます。

## ROLLBACK後のデータの状態: 例

```
DELETE FROM test;  
25,000 rows deleted.  
  
ROLLBACK;  
Rollback complete.  
  
DELETE FROM test WHERE id = 100;  
1 row deleted.  
  
SELECT * FROM test WHERE id = 100;  
No rows selected.  
  
COMMIT;  
Commit complete.
```

ORACLE

Copyright © 2007, Oracle. All rights reserved.

### ROLLBACK後のデータの状態: 例

TEST表から1レコードの削除を試みる際に、誤って表を空にしてしまう場合があります。しかし、この間違いを修正し、適切な文を再発行し、データ変更を確定することができます。

## 文レベル・ロールバック

- 実行中に1つのDML文でエラーが発生した場合、その文のみがロールバックされます。
- Oracleサーバーでは、暗黙的セーブポイントが実装されています。
- その他の変更はすべて保持されます。
- ユーザーは、COMMIT文またはROLLBACK文を実行して、明示的にトランザクションを終了する必要があります。

ORACLE

Copyright © 2007, Oracle. All rights reserved.

### 文レベル・ロールバック

文の実行エラーが検出された場合には、暗黙的なロールバックによりトランザクションの一部を破棄することができます。トランザクションの実行中に1つのDML文でエラーが発生した場合、文レベルのロールバックによってその実行内容は元に戻されますが、そのトランザクションにおいて、それ以前のDML文で実行された変更は破棄されません。それらの変更は、ユーザーが明示的にコミットまたはロールバックすることができます。

Oracleサーバーでは、DDL文の前と後に暗黙的なコミットが発行されます。そのため、DDL文が正しく実行されない場合であっても、サーバーがコミットを発行済みであるため、前の文をロールバックすることはできません。

トランザクションを明示的に終了するには、COMMIT文またはROLLBACK文を実行します。

## 章の講義項目

- 表への新しい行の追加
  - INSERT文
- 表内のデータの変更
  - UPDATE文
- 表からの行の削除
  - DELETE文
  - TRUNCATE文
- COMMIT、ROLLBACKおよびSAVEPOINTによるデータベース・トランザクション制御
- 読取り一貫性
- SELECT文のFOR UPDATE句

ORACLE

Copyright © 2007, Oracle. All rights reserved.

## 読取り一貫性

- 読取り一貫性により、常に、一貫性のあるデータのビューが保証されます。
- あるユーザーによる変更が別のユーザーによる変更と競合することはありません。
- 読取り一貫性により、同一データの操作に関して、次のことが保証されます。
  - リーダーの操作はライターの操作と無関係です。
  - ライターの操作はリーダーの操作と無関係です。
  - ライターは別のライターの操作終了まで待機する必要があります。

ORACLE

Copyright © 2007, Oracle. All rights reserved.

### 読取り一貫性

データベースのユーザーは、次の2種類の方法でデータベースにアクセスします。

- 読取り操作 (SELECT文)
- 書込み操作 (INSERT文、UPDATE文、DELETE文)

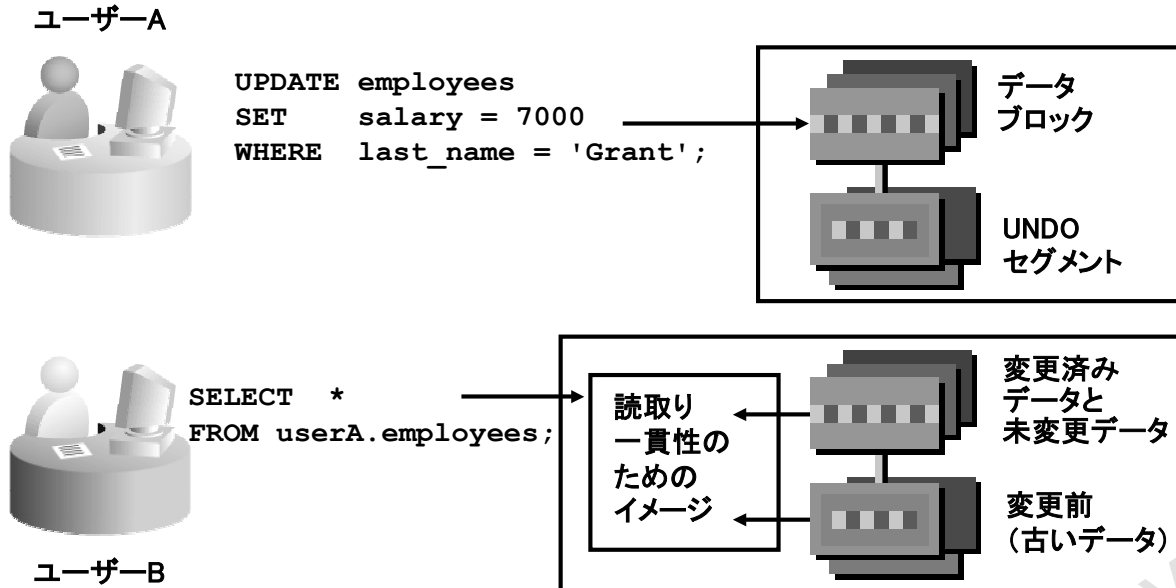
次のように動作するためには、読取り一貫性が必要になります。

- データベースのリーダーおよびライターは、一貫性のあるデータのビューが保証される
- リーダーに変更途中のデータが表示されない
- ライターは、データベースの変更が一貫した方法で実行されることが保証される
- あるライターによる変更が、別のライターによる進行中の変更を阻害すること、または変更内容の競合が発生することがない

読取り一貫性の目的は、DML操作が開始される前の、最後のコミット時点の状態のデータが各ユーザーに表示されるように保証することです。

**注意:** 同一ユーザーが別のセッションとしてログインすることは可能です。それぞれのセッションは、同一ユーザーの場合であっても、前述の方法で読取り一貫性が維持されます。

# 読取り一貫性の実装



ORACLE

Copyright © 2007, Oracle. All rights reserved.

## 読取り一貫性の実装

読取り一貫性は自動的に実装されます。UNDOセグメント内にデータベースの部分的なコピーが維持されます。読取り一貫性のためのイメージは、表内のコミット済みデータと、コミットされていない現在変更中データの変更前のデータ(UNDOセグメントのデータ)で構成されます。

データベースで挿入、更新または削除の操作が実行されると、Oracleサーバーでは、変更前に該当データのコピーが作成され、UNDOセグメントに書き込まれます。

変更を発行したユーザー以外のすべてのリーダーには、データベースの変更開始前の状態が表示されます。つまり、UNDOセグメントにあるデータのスナップショットが表示されています。

変更がデータベースにコミットされる前は、データを変更中のユーザーに対してのみ、変更済みのデータベースの内容が表示されます。他のすべてのユーザーには、UNDOセグメント内のスナップショットが表示されます。これによって、データのリーダーには現在変更途中のデータではなく、一貫性のあるデータの読取りが保証されます。

DML文がコミットされると、データベースの変更が表示可能になり、コミット後にSELECT文を発行したすべてのユーザーに表示されます。UNDOセグメント内の古いデータの占有スペースは解放され、再利用が可能となります。

トランザクションがロールバックされた場合、次のように変更が元に戻されます。

- UNDOセグメント内にある変更前の元のデータが、表に上書きされて戻されます。
- すべてのユーザーには、データベースのトランザクション開始前の状態が表示されます。

## 章の講義項目

- 表への新しい行の追加
  - INSERT文
- 表内のデータの変更
  - UPDATE文
- 表からの行の削除
  - DELETE文
  - TRUNCATE文
- COMMIT、ROLLBACKおよびSAVEPOINTによるデータベース・トランザクション制御
- 読取り一貫性
- SELECT文のFOR UPDATE句

ORACLE

Copyright © 2007, Oracle. All rights reserved.



## SELECT文のFOR UPDATE句

- EMPLOYEES表で、job\_idがSA\_REPである行がロックされます。

```
SELECT employee_id, salary, commission_pct, job_id
FROM employees
WHERE job_id = 'SA_REP'
FOR UPDATE
ORDER BY employee_id;
```

- ロックは、ROLLBACKまたはCOMMITを発行した場合にのみ解除されます。
- SELECT文でロックを試みた行が別のユーザーによってすでにロックされている場合、データベースはその行が利用可能になるまで待機し、その後SELECT文の結果を戻します。

ORACLE

Copyright © 2007, Oracle. All rights reserved.

### SELECT文のFOR UPDATE句

データベースに対してSELECT文を発行していくつかのレコードを問い合わせる場合、選択された行に対するロックは行われません。通常、これはロックするレコード数を常に絶対最小値に維持する(デフォルト)ために必要な措置です。コミットされていない変更中のレコードのみがロックされます。ロックされていても、他のユーザーはこれらのレコードの変更前の状態(データの変更前イメージ)を読み取ることができます。一方、プログラム内でレコードのセットを変更する前にロックする必要がある場合もあります。Oracleでは、このようなロックを実行するために、SELECT文のFOR UPDATE句が提供されています。

SELECT...FOR UPDATE文を発行すると、リレーショナル・データベース管理システム(RDBMS)では、SELECT文で特定されるすべての行において行レベルの排他ロックが自動的に取得され、発行者の変更専用にレコードが保持されます。ROLLBACKまたはCOMMITを実行するまで、他のユーザーはこれらのレコードを変更できません。

オプションのキーワードのNOWAITをFOR UPDATE句に付けると、別のユーザーによって表がロックされている場合にはその解放を待機しないようにOracleサーバーに対して指示できます。この場合、即座にプログラムまたはSQL Developer環境に制御が戻されるため、ロックを再度試行するまでの間、他の作業を実行することも、そのまま待機することもできます。NOWAIT句を使用しない場合、ロックを実行したユーザーによってCOMMITコマンドまたはROLLBACKコマンドが発行され、ロックが解除されて表が利用可能になるまで、プロセスは待機状態になります。

## FOR UPDATE句: 例

- SELECT文のFOR UPDATE句は、複数の表に対して使用できます。

```
SELECT e.employee_id, e.salary, e.commission_pct
FROM employees e JOIN departments d
USING (department_id)
WHERE job_id = 'ST_CLERK'
AND location_id = 1500
FOR UPDATE
ORDER BY e.employee_id;
```

- EMPLOYEES表とDEPARTMENTS表の両方の行がロックされます。
- FOR UPDATE OF *column\_name*を使用して変更対象の列を限定すると、その特定の表の行のみがロックされます。

ORACLE

Copyright © 2007, Oracle. All rights reserved.

### FOR UPDATE句: 例

スライドの例では、EMPLOYEES表でJOB\_IDにST\_CLERKが設定され、LOCATION\_IDに1500が設定されている行がロックされ、DEPARTMENTS表でLOCATION\_IDに1500が設定されている部門の行がロックされます。

FOR UPDATE OF *column\_name*を使用すると、変更対象の列を限定することができます。FOR UPDATE句のOFリストを指定したとしても、選択した行のこれらの列しか変更できないわけではありません。ロックはすべての行に対して適用されます。問合せでFOR UPDATEのみを指定し、OFキーワードの後に列を1つ以上指定しない場合は、FROM句に指定したすべての表で、特定されたすべての行がデータベースによってロックされます。

次の文では、EMPLOYEES表でLOCATION\_IDが1500で、ST\_CLERKの値を含む行のみロックされます。DEPARTMENTS表の行はロックされません。

```
SELECT e.employee_id, e.salary, e.commission_pct
FROM employees e JOIN departments d
USING (department_id)
WHERE job_id = 'ST_CLERK' AND location_id = 1500
FOR UPDATE OF e.salary
ORDER BY e.employee_id;
```

### FOR UPDATE句: 例(続き)

次の例では、行が使用可能になるのを5秒間待機したら、制御をユーザーに戻すようにデータベースに指示します。

```
SELECT employee_id, salary, commission_pct, job_id
FROM employees
WHERE job_id = 'SA_REP'
FOR UPDATE WAIT 5
ORDER BY employee_id;
```

Oracle Internal & Oracle Academy  
Use Only

## まとめ

この章では、次の文の使用方法について学習しました。

関数	説明
INSERT	表に新しい行を追加します。
UPDATE	表内の既存の行を変更します。
DELETE	表から既存の行を削除します。
TRUNCATE	表からすべての行を削除します。
COMMIT	保留中の変更をすべて確定します。
SAVEPOINT	セーブポイント・マーカーへのロールバックに使用されます。
ROLLBACK	保留中のデータ変更をすべて破棄します。
SELECT文の FOR UPDATE句	SELECT問合せで特定される行をロックします。

ORACLE

Copyright © 2007, Oracle. All rights reserved.

### まとめ

この章では、INSERT文、UPDATE文、DELETE文およびTRUNCATE文を使用してOracleデータベースのデータを操作する方法と、COMMIT文、SAVEPOINT文およびROLLBACK文を使用してデータの変更を制御する方法について学習しました。また、SELECT文のFOR UPDATE句を使用して、変更対象の行のみをロックする方法についても学習しました。

Oracleサーバーでは、常に、一貫性のあるデータのビューが保証されます。

## 演習9: 概要

この演習では次の項目について説明しています。

- 表への行の挿入
- 表内の行の更新および削除
- トランザクションの制御

ORACLE

Copyright © 2007, Oracle. All rights reserved.

### 演習9: 概要

この演習では、MY\_EMPLOYEE表への行の追加、この表のデータの更新と削除、およびトランザクションの制御を行います。また、MY\_EMPLOYEE表を作成するスクリプトを実行します。

## 演習9

HR部門では、従業員データの挿入、更新および削除を行うSQL文の作成を必要としています。HR部門にSQL文を提出する前のプロトタイプとしてMY\_EMPLOYEE表を使用します。

**注意:** すべてのDML文について、「Run Script」アイコンを使用して(または[F5]を押して)問合せを実行します。これにより、「Script Output」タブ・ページにフィードバック・メッセージが表示されます。SELECT問合せの場合は、さらに「Execute Statement」アイコンを使用するか、[F9]を押すと、「Results」タブ・ページに書式設定された出力が表示されます。

### MY\_EMPLOYEE表へのデータの挿入

1. lab\_09\_01.sqlスクリプトの文を実行して、この演習で使用するMY\_EMPLOYEE表を作成します。
2. MY\_EMPLOYEE表の構造を記述して、列の名前を特定します。

DESCRIBE MY_EMPLOYEE		
Name	Null	Type
-----		
ID	NOT NULL	NUMBER(4)
LAST_NAME		VARCHAR2(25)
FIRST_NAME		VARCHAR2(25)
USERID		VARCHAR2(8)
SALARY		NUMBER(9,2)

3. 次のサンプルデータから、データの最初の行をMY\_EMPLOYEE表に追加するINSERT文を作成します。INSERT句には列リストを指定しません。まだ、すべての行は入力しないでください。

ID	LAST_NAME	FIRST_NAME	USERID	SALARY
1	Patel	Ralph	rpatel	895
2	Dancs	Betty	bdancs	860
3	Biri	Ben	bbiri	1100
4	Newman	Chad	cnewman	750
5	Ropeburn	Audrey	aropebur	1550

4. 前述のリストのサンプル・データの2行目をMY\_EMPLOYEE表に移入します。今度は、INSERT句に列リストを明示的に指定してください。

## 演習9(続き)

5. 表への追加結果を確認します。

	ID	LAST_NAME	FIRST_NAME	USERID	SALARY
1	1	Patel	Ralph	rpatel	895
2	2	Dancs	Betty	bdancs	860

6. 残りの行をMY\_EMPLOYEE表にロードするINSERT文を、動的に再利用可能なスクリプト・ファイルに記述します。スクリプトでは、すべての列 (ID、LAST\_NAME、FIRST\_NAME、USERIDおよびSALARY) の入力をユーザーに求める必要があります。このスクリプトをlab\_09\_06.sqlファイルに保存します。
7. 作成したスクリプトのINSERT文を実行して、ステップ3で示したサンプル・データの次の2行を表に移入します。
8. 表への追加結果を確認します。

	ID	LAST_NAME	FIRST_NAME	USERID	SALARY
1	1	Patel	Ralph	rpatel	895
2	2	Dancs	Betty	bdancs	860
3	3	Biri	Ben	bbiri	1100
4	4	Newman	Chad	cnewman	750

9. データの追加を確定します。

## MY\_EMPLOYEE表内のデータの更新と削除

10. 従業員3の姓をDrexlerに変更します。
11. 給与が\$900未満の従業員すべての給与を\$1,000に変更します。
12. 表の変更結果を確認します。

	ID	LAST_NAME	FIRST_NAME	USERID	SALARY
1	1	Patel	Ralph	rpatel	1000
2	2	Dancs	Betty	bdancs	1000
3	3	Drexler	Ben	bbiri	1100
4	4	Newman	Chad	cnewman	1000

13. MY\_EMPLOYEE表からBetty Dancsを削除します。
14. 表の変更結果を確認します。

	ID	LAST_NAME	FIRST_NAME	USERID	SALARY
1	1	Patel	Ralph	rpatel	1000
2	3	Drexler	Ben	bbiri	1100
3	4	Newman	Chad	cnewman	1000

## 演習9(続き)

15. 保留中の変更をすべてコミットします。

### MY\_EMPLOYEE表のデータ・トランザクションの制御

16. ステップ6で作成したスクリプトの文を使用して、ステップ3で示したサンプル・データの最終行を表に移入します。スクリプトの文を実行します。

17. 表への追加結果を確認します。

	ID	LAST_NAME	FIRST_NAME	USERID	SALARY
1	1	Patel	Ralph	rpatel	1000
2	3	Drexler	Ben	bbiri	1100
3	4	Newman	Chad	cnewman	1000
4	5	Ropeburn	Audrey	aropebur	1550

18. このトランザクション処理内で、中間点にマーカーを設定します。

19. MY\_EMPLOYEE表からすべての行を削除します。

20. 表が空であることを確認します。

21. 最後のDELETE操作を破棄します。ただし、それ以前のINSERT操作は破棄しないでください。

22. 新しい行がそのまま存在することを確認します。

	ID	LAST_NAME	FIRST_NAME	USERID	SALARY
1	1	Patel	Ralph	rpatel	1000
2	3	Drexler	Ben	bbiri	1100
3	4	Newman	Chad	cnewman	1000
4	5	Ropeburn	Audrey	aropebur	1550

23. データの追加を確定します。

時間があるときは、次の演習問題に進みます。

24. lab\_09\_06.sqlスクリプトを変更し、名の最初の文字と姓の最初の7文字を連結することによってUSERIDが自動的に生成されるようにします。生成されるUSERIDは必ず小文字にしてください。これで、スクリプトでUSERIDの入力をユーザーに求める必要はなくなります。このスクリプトをlab\_09\_24.sqlというファイルに保存します。

25. スクリプトlab\_09\_24.sqlを実行して、次のレコードを挿入します。

ID	LAST_NAME	FIRST_NAME	USERID	SALARY
6	Anthony	Mark	manthony	1230

26. 新しい行が追加され、正しいUSERIDが含まれていることを確認します。

ID	LAST_NAME	FIRST_NAME	USERID	SALARY
6	Anthony	Mark	manthony	1230



# DDL文を使用した表の作成および管理

10

ORACLE

Copyright © 2007, Oracle. All rights reserved.

Oracle Internal & Oracle Academy  
Use Only

## 目的

この章を終えると、次のことができるようになります。

- 主要なデータベース・オブジェクトの分類
- 表構造の説明
- 列で使用可能なデータ型のリスト
- 単純な表の作成
- 表の作成時における制約の作成方法の説明
- スキーマ・オブジェクトの機能の説明

ORACLE

Copyright © 2007, Oracle. All rights reserved.

### 目的

この章では、データ定義言語 (DDL) 文について説明します。単純な表を作成、変更および削除する基本的な方法について学習できます。DDLで使用可能なデータ型を示し、スキーマの概念を紹介します。制約についてもこの章で説明します。DML操作中の制約違反によって生成される例外メッセージも示し、それぞれ説明します。

## 章の講義項目

- データベース・オブジェクト
  - ネーミング規則
- CREATE TABLE文:
  - 別のユーザーの表にアクセス
  - DEFAULTオプション
- データ型
- 制約の概要: NOT NULL、PRIMARY KEY、FOREIGN KEY、CHECK制約
- 副問合せを使用した表の作成
- ALTER TABLE
  - 読取り専用の表
- DROP TABLE文

ORACLE

Copyright © 2007, Oracle. All rights reserved.

# データベース・オブジェクト

オブジェクト	説明
表	行で構成される、記憶域の基本単位。
ビュー	1つ以上の表からのデータのサブセットを論理的に表します。
順序	数値を生成します。
索引	一部の問合せのパフォーマンスが向上します。
シノニム	オブジェクトに別名を付けます。

ORACLE

Copyright © 2007, Oracle. All rights reserved.

## データベース・オブジェクト

Oracle Databaseには、複数のデータ構造を含めることができます。データベース開発の構築段階でデータ構造を作成できるように、それぞれの構造をデータベース設計時に示す必要があります。

- **表:** データの格納
- **ビュー:** 1つ以上の表からのデータのサブセット
- **順序:** 数値の生成
- **索引:** 一部の問合せのパフォーマンス向上
- **シノニム:** オブジェクトへの別名の指定

### Oracleの表構造

- 表は、ユーザーがデータベースを使用中であっても、いつでも作成できます。
- 表のサイズを指定する必要はありません。サイズは、データベースに全体として割り当てられる領域の量によって、最終的に定義されます。ただし、表が長期的に使用する領域の量を見積もることが重要です。
- 表構造はオンラインで変更できます。

**注意:** データベース・オブジェクトは他にもありますが、このコースでは説明していません。

# ネーミング規則

表および列の名前には、次の規則があります。

- 文字で始まります。
- 長さは1～30文字です。
- 使用できるのはA～Z、a～z、0～9、\_、\$および#のみです。
- 同じユーザーが所有する別のオブジェクトと重複する名前は使用できません。
- Oracleサーバー用の予約語は使用できません。

ORACLE

Copyright © 2007, Oracle. All rights reserved.

## ネーミング規則

データベースの表と列には、Oracle Databaseオブジェクトの標準のネーミング規則に従って名前を付けます。

- 表と列の名前は、文字で始まり、1～30文字の範囲にする必要があります。
- 名前で使用できるのは、A～Z、a～z、0～9、\_(アンダースコア)、\$および#(使用できるが推奨されない)だけです。
- 同じOracleサーバー・ユーザーが所有する別のオブジェクトと重複する名前は使用できません。
- Oracleサーバーの予約語は使用できません。
  - オブジェクトの名前を表すために、引用識別子を使用することもできます。引用識別子の先頭と末尾には、二重引用符(“)を付けます。引用識別子を使用してスキーマ・オブジェクトに名前を付けた場合は、そのオブジェクトを参照するときに必ず二重引用符を使用する必要があります。引用識別子には予約語を使用できますが、お薦めできません。

## ネーミング・ガイドライン

表およびその他のデータベース・オブジェクトには、わかりやすい名前を付けてください。

**注意:** 名前では大文字と小文字は区別されません。たとえば、EMPLOYEESはeMPloyeesまたはeMpLOYEESと同じ名前として扱われます。ただし、引用識別子では大文字と小文字が区別されます。

詳細は、『Oracle Database SQL言語リファレンス11gリリース1(11.1)』のスキーマ・オブジェクト名および修飾子に関する節を参照してください。

## 章の講義項目

- データベース・オブジェクト
  - ネーミング規則
- CREATE TABLE文:
  - 別のユーザーの表にアクセス
  - DEFAULTオプション
- データ型
- 制約の概要: NOT NULL、PRIMARY KEY、FOREIGN KEY、CHECK制約
- 副問合せを使用した表の作成
- ALTER TABLE
  - 読取り専用の表
- DROP TABLE文

ORACLE

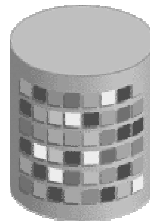
Copyright © 2007, Oracle. All rights reserved.

# CREATE TABLE文

- 次のものがが必要です。
  - CREATE TABLE権限
  - 記憶域

```
CREATE TABLE [schema.] table  
    (column datatype [DEFAULT expr] [, ...]);
```

- 次のものを指定します。
  - 表の名前
  - 列の名前、列のデータ型、および列のサイズ



ORACLE

Copyright © 2007, Oracle. All rights reserved.

## CREATE TABLE文

データを格納する表を作成するには、SQLのCREATE TABLE文を実行します。この文は、Oracle Database構造の作成、変更または削除に使用する、SQL文のサブセットであるDDL文の1つです。これらのDDL文は、データベースに即座に適用され、データ・ディクショナリにも情報が記録されます。

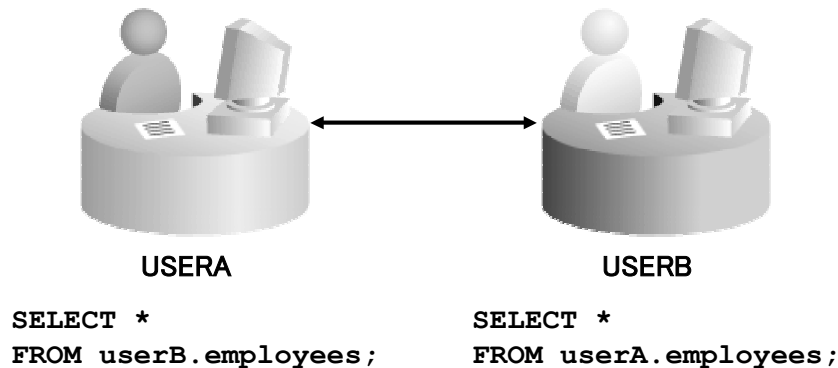
表を作成するには、ユーザーはCREATE TABLE権限を持ち、オブジェクトを作成するための記憶域がなければなりません。ユーザーへの権限付与は、データベース管理者 (DBA) がデータ制御言語 (DCL) 文を使用して行います。

構文の内容

<i>schema</i>	所有者の名前と同一
<i>table</i>	表の名前
DEFAULT <i>expr</i>	INSERT文で値が省略されている場合のデフォルト値
<i>column</i>	列の名前
<i>datatype</i>	列のデータ型およびデータ長

## 別のユーザーの表の参照

- ユーザーのスキーマ内には、他のユーザーに属する表は存在しません。
- このような表には、接頭辞として所有者の名前を使用する必要があります。



ORACLE

Copyright © 2007, Oracle. All rights reserved.

### 別のユーザーの表の参照

スキーマは、データまたはスキーマ・オブジェクトの論理構造のコレクションです。スキーマはデータベース・ユーザーによって所有され、そのユーザーと同じ名前を持ちます。ユーザーはそれぞれ1つのスキーマを所有します。

スキーマ・オブジェクトは、SQLを使用して作成および操作可能で、表、ビュー、シノニム、順序、ストアド・プロシージャ、索引、クラスタおよびデータベース・リンクを含めることができます。

ユーザーが自分の所有ではない表を使用する場合は、表の接頭辞として所有者の名前を付ける必要があります。たとえば、USERAおよびUSERBという名前のスキーマがあり、そのどちらにもEMPLOYEES表がある場合、USERAがUSERBに属するEMPLOYEES表にアクセスするには、USERAは表名の接頭辞としてスキーマ名を付ける必要があります。

```
SELECT *  
FROM   userb.employees;
```

USERAが所有するEMPLOYEES表にUSERBがアクセスするには、USERBは表名の接頭辞としてスキーマ名を付ける必要があります。

```
SELECT *  
FROM   usera.employees;
```



## DEFAULTオプション

- 挿入時の列のデフォルト値を指定します。

```
... hire_date DATE DEFAULT SYSDATE, ...
```

- 有効な値はリテラル値、式またはSQL関数です。
- 別の列の名前または疑似列は無効な値です。
- デフォルトのデータ型は列のデータ型に一致していなければなりません。

```
CREATE TABLE hire_dates  
  (id          NUMBER(8),  
   hire_date DATE DEFAULT SYSDATE);  
CREATE TABLE succeeded.
```

ORACLE

Copyright © 2007, Oracle. All rights reserved.

### DEFAULTオプション

表を定義する際に、DEFAULTオプションを使用して、列にデフォルト値が入力されるように指定できます。このオプションによって、列に対する値がない状態で行が挿入されたときに、列にNULL値が入力されるのを防ぐことができます。デフォルト値には、リテラル、式またはSQL関数（SYSDATEまたはUSERなど）を使用できますが、デフォルト値に別の列または疑似列の名前（NEXTVALまたはCURRVALなど）を使用することはできません。デフォルトの式は、列のデータ型に一致する必要があります。

次の例を検討してみます。

```
INSERT INTO hire_dates values(45, NULL);
```

この文では、デフォルト値ではなくNULL値が挿入されます。

```
INSERT INTO hire_dates(id) values(35);
```

この文では、HIRE\_DATE列にSYSDATEが挿入されます。

**注意:** これらのDDL文を実行するには、SQL Developerで「Run Script」アイコンをクリックするか、[F5]を押します。「Script Output」タブ・ページにフィードバック・メッセージが表示されます。

## 表の作成

- 表を作成します。

```
CREATE TABLE dept
  (deptno      NUMBER(2),
   dbname      VARCHAR2(14),
   loc         VARCHAR2(13),
   create_date DATE DEFAULT SYSDATE);
```

```
CREATE TABLE succeeded.
```

- 表の作成を確認します。

```
DESCRIBE dept
```

DESCRIBE dept		
Name	Null	Type
-----		
DEPTNO		NUMBER(2)
DNAME		VARCHAR2(14)
LOC		VARCHAR2(13)
CREATE_DATE		DATE

ORACLE

Copyright © 2007, Oracle. All rights reserved.

### 表の作成

このスライドの例では、DEPTNO、DNAME、LOCおよびCREATE\_DATEの4つの列でDEPT表を作成します。CREATE\_DATE列にはデフォルト値があります。INSERT文に値を指定しない場合は、自動的にシステム日付が挿入されます。

表が作成されたことを確認するには、DESCRIBEコマンドを実行します。

表を作成するのはDDL文であるため、この文を実行すると自動コミットが発生します。

## 章の講義項目

- データベース・オブジェクト
  - ネーミング規則
- CREATE TABLE文:
  - 別のユーザーの表にアクセス
  - DEFAULTオプション
- データ型
- 制約の概要: NOT NULL、PRIMARY KEY、FOREIGN KEY、CHECK制約
- 副問合せを使用した表の作成
- ALTER TABLE
  - 読取り専用の表
- DROP TABLE文

ORACLE

Copyright © 2007, Oracle. All rights reserved.

# データ型

データ型	説明
VARCHAR2( <i>size</i> )	可変長文字データ
CHAR( <i>size</i> )	固定長文字データ
NUMBER( <i>p,s</i> )	可変長数値データ
DATE	日付と時刻の値
LONG	可変長文字データ(最大2GB)
CLOB	文字データ(最大4GB)
RAWおよび LONG RAW	RAWバイナリ・データ
BLOB	バイナリ・データ(最大4GB)
BFILE	外部ファイルに格納されるバイナリ・データ(最大4GB)
ROWID	表内の行の一意のアドレスを表す、Base64の記数法

ORACLE

Copyright © 2007, Oracle. All rights reserved.

## データ型

表の列を識別するには、その列のデータ型を指定する必要があります。使用できるデータ型にはいくつかの種類があります。

データ型	説明
VARCHAR2( <i>size</i> )	可変長文字データ。最大の <i>size</i> を指定する必要があります。 <i>size</i> の最小値は1、最大値は4,000です。
CHAR [( <i>size</i> )]	<i>size</i> に指定したバイト数の固定長文字データ。 <i>size</i> のデフォルトおよび最小値は1、最大値は2,000です。
NUMBER [( <i>p,s</i> )]	精度 <i>p</i> および位取り <i>s</i> を指定した数値。精度は10進数字の総桁数であり、位取りは小数点以下の桁数です。精度の範囲は1～38、スケールの範囲は-84～127です。
DATE	紀元前4712年1月1日～紀元9999年12月31日の範囲における、直近の秒数に対する日付と時刻の値。
LONG	可変長文字データ(最大2GB)
CLOB	文字データ(最大4GB)

## データ型(続き)

データ型	説明
RAW( <i>size</i> )	長さが <i>size</i> の RAW バイナリ・データ。最大の <i>size</i> を指定する必要があります。 <i>size</i> の最大値は 2,000 です。
LONG RAW	可変長 RAW バイナリ・データ (最大 2GB)。
BLOB	バイナリ・データ (最大 4GB)。
BFILE	外部ファイルに格納されるバイナリ・データ (最大 4GB)。
ROWID	表内の行の一意のアドレスを表す、BASE64 の記数法。

## ガイドライン

- 副問合せを使用して表を作成した場合、LONG 列はコピーされません。
- LONG 列を GROUP BY または ORDER BY 句に含めることはできません。
- 1 つの表で利用できる LONG 列は 1 つだけです。
- LONG 列に制約を定義することはできません。
- LONG 列よりも CLOB 列が適している場合があります。

Oracle Internal & Oracle Academy  
Use Only

# 日時データ型

使用できる日時データ型には、次のような種類があります。

データ型	説明
TIMESTAMP	秒の小数部を含む日付
INTERVAL YEAR TO MONTH	期間を年および月として格納します。
INTERVAL DAY TO SECOND	期間を日付、時、分、秒として格納します。



ORACLE

Copyright © 2007, Oracle. All rights reserved.

## 日時データ型

データ型	説明
TIMESTAMP	秒の小数部を持つ日付として時間を格納できます。DATEデータ型の年、月、日、時、分および秒の値に加えて、秒の小数部の値を格納します。 このデータ型には、WITH TIMEZONE、WITH LOCALTIMEZONEなどいくつかのバリエーションがあります。
INTERVAL YEAR TO MONTH	時間を年と月の期間として格納できます。年と月のみが重要である場合に、2つの日時の値の違いを表すために使用します。
INTERVAL DAY TO SECOND	時間を日付、時、分、秒の期間として格納できます。2つの日時の値の正確な違いを表すために使用します。

**注意:** これらの日時データ型は、Oracle9i以上のリリースで使用できます。日時データ型の詳細については、『Oracle Database 11g: 入門 SQL 基礎 II Ed 1』コースの様々なタイムゾーンのデータの管理に関する節で説明しています。

また、日時データ型の詳細は、『Oracle Database SQL言語リファレンス11gリリース1(11.1)』のTIMESTAMPデータ型、INTERVAL YEAR TO MONTHデータ型、INTERVAL DAY TO SECONDデータ型に関する節を参照してください。

## 章の講義項目

- データベース・オブジェクト
  - ネーミング規則
- CREATE TABLE文:
  - 別のユーザーの表にアクセス
  - DEFAULTオプション
- データ型
- 制約の概要: NOT NULL、PRIMARY KEY、FOREIGN KEY、CHECK制約
- 副問合せを使用した表の作成
- ALTER TABLE
  - 読取り専用の表
- DROP TABLE文

ORACLE

Copyright © 2007, Oracle. All rights reserved.

## 制約の設定

- 制約により、表レベルでルールが適用されます。
- 依存関係がある場合、制約によって表の削除が防止されます。
- 有効な制約の種類は、次のとおりです。
  - NOT NULL
  - UNIQUE
  - PRIMARY KEY
  - FOREIGN KEY
  - CHECK



ORACLE

Copyright © 2007, Oracle. All rights reserved.

### 制約

Oracleサーバーでは制約を使用することで、表への無効なデータ入力を防ぎます。

制約は、次のような場合に使用できます。

- 表で行が挿入、更新または削除されたときに、必ず表のデータにルールを適用する場合。この操作が正常に終了するには、制約条件を満たしている必要があります。
- 他の表からの依存性があるときに、表が削除されないようにする場合。
- Oracle DeveloperなどのOracleツールに対するルールを提供する場合。

### データの整合性制約

制約	説明
NOT NULL	列にNULL値を入力できないことを指定します。
UNIQUE	列または列の組合せに対して、表内のすべての行について値が一意でなければならないことを指定します。
PRIMARY KEY	表の各行を一意に識別します。
FOREIGN KEY	1つの表の値が別の表の値に一致するように、特定の列と参照先の表の列との参照整合性を確立し、適用します。
CHECK	条件を満たす必要があることを指定します。



## 制約のガイドライン

- 制約には名前を付けることができます。名前を付けない場合は、OracleサーバーがSYS\_C*n*の形式で名前を作成します。
- 制約は次のいずれかの時点で作成します。
  - 表の作成と同時
  - 表の作成後
- 制約は列レベルまたは表レベルで定義します。
- 制約はデータ・ディクショナリに表示されます。

ORACLE

Copyright © 2007, Oracle. All rights reserved.

### 制約のガイドライン

制約はすべてデータ・ディクショナリに格納されます。制約に意味のある名前を付けると、参照が簡単になります。制約には、オブジェクトの標準のネーミング規則に従って名前を付け、同じユーザーが所有する別のオブジェクトと同じ名前は使用できません。制約に名前を付けない場合は、OracleサーバーがSYS\_C*n*の形式で名前を生成します。*n*は、制約の名前を一意にするための整数です。

制約は、表の作成時と作成後のどちらでも定義できます。制約は列レベルまたは表レベルで定義できます。表レベルの制約と列レベルの制約は、機能的には同じです。

詳細は、『Oracle Database SQL言語リファレンス11*g*リリース1(11.1)』の制約に関する節を参照してください。

# 制約の定義

- 構文:

```
CREATE TABLE [schema.] table
  (column datatype [DEFAULT expr]
   [column_constraint],
   ...
   [table_constraint] [, ...]);
```

- 列レベル制約の構文:

```
column [CONSTRAINT constraint_name] constraint_type,
```

- 表レベル制約の構文:

```
column, ...
[CONSTRAINT constraint_name] constraint_type
(column, ...),
```

ORACLE

Copyright © 2007, Oracle. All rights reserved.

## 制約の定義

このスライドでは、表の作成時に制約を定義する構文を示しています。制約は、列レベルまたは表レベルのどちらでも作成できます。列レベルで定義する制約は、列の定義時に設定します。表レベルの制約は表の定義の最後に定義し、その制約が関係する1つ以上の列をカッコで囲んで参照する必要があります。列レベルの制約と表レベルの制約の違いは、主に構文上の点であり、機能上は同じです。

NOT NULL制約は列レベルで定義する必要があります。

複数の列に適用される制約は、表レベルで定義する必要があります。

構文の内容

<i>schema</i>	所有者の名前と同一
<i>table</i>	表の名前
DEFAULT <i>expr</i>	INSERT文で値が省略されている場合のデフォルト値
<i>column</i>	列の名前
<i>datatype</i>	列のデータ型およびデータ長
<i>column_constraint</i>	列の定義の一部としての整合性制約
<i>table_constraint</i>	表の定義の一部としての整合性制約

## 制約の定義

- 列レベルの制約の例:

```
CREATE TABLE employees (  
  employee_id NUMBER(6)  
  CONSTRAINT emp_emp_id_pk PRIMARY KEY,  
  first_name VARCHAR2(20),  
  ...);
```

1

- 表レベルの制約の例:

```
CREATE TABLE employees (  
  employee_id NUMBER(6),  
  first_name VARCHAR2(20),  
  ...  
  job_id VARCHAR2(10) NOT NULL,  
  CONSTRAINT emp_emp_id_pk  
    PRIMARY KEY (EMPLOYEE_ID));
```

2

ORACLE

Copyright © 2007, Oracle. All rights reserved.

### 制約の定義(続き)

通常、制約は表と同時に作成します。制約は表の作成後に追加することも、一時的に無効にすることもできます。

スライドの例では、どちらもEMPLOYEES表のEMPLOYEE\_ID列に主キー制約を作成しています。

1. 最初の例では、列レベルの構文を使用して制約を定義しています。
2. 2番目の例では、表レベルの構文を使用して制約を定義しています。

主キー制約の詳細は、この章で後述します。

## NOT NULL制約

列でNULL値が許可されないことを保証:

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	HIRE_DATE	JOB_ID	COMMISSION_PCT
100	Steven	King	SKING	17-JUN-87	AD_PRES	(null)
101	Neena	Kochhar	NKOCHHAR	21-SEP-89	AD_VP	(null)
102	Lex	De Haan	LDEHAAN	13-JAN-93	AD_VP	(null)
103	Alexander	Hunold	AHUNOLD	03-JAN-90	IT_PROG	(null)
104	Bruce	Ernst	BERNST	21-MAY-91	IT_PROG	(null)
107	Diana	Lorentz	DLORENTZ	07-FEB-99	IT_PROG	(null)
124	Kevin	Mourgos	KMOURGOS	16-NOV-99	ST_MAN	(null)
141	Trenna	Rajs	TRAJS	17-OCT-95	ST_CLERK	(null)
142	Curtis	Davies	CDAVIES	29-JAN-97	ST_CLERK	(null)
143	Randall	Matos	RMATOS	15-MAR-98	ST_CLERK	(null)
144	Peter	Vargas	PVARGAS	09-JUL-98	ST_CLERK	(null)
149	Eleni	Zlotkey	EZLOTKEY	29-JAN-00	SA_MAN	0.2
174	Ellen	Abel	EABEL	11-MAY-96	SA_REP	0.3

... ↑  
**NOT NULL制約**  
(主キーによりNOT NULL  
制約が強制される)

↑  
**NOT NULL  
制約**

↑  
**NOT NULL制約なし**  
(この列の行にはNULL値を  
含めることができる)

ORACLE

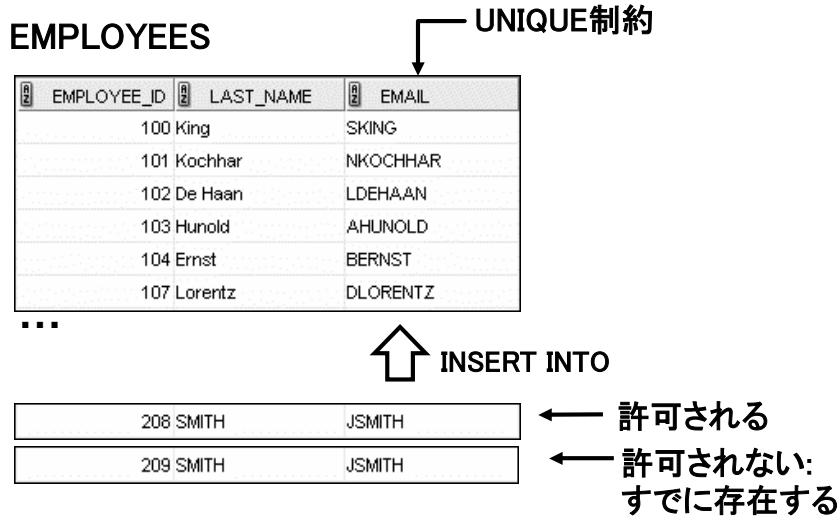
Copyright © 2007, Oracle. All rights reserved.

### NOT NULL制約

NOT NULL制約を使用すると、列にNULL値が含まれないことが保証されます。NOT NULL制約を適用しない列には、デフォルトでNULL値を含めることができます。NOT NULL制約は列レベルで定義する必要があります。EMPLOYEES表のEMPLOYEE\_ID列は主キーとして定義されているため、NOT NULL制約を継承します。LAST\_NAME、EMAIL、HIRE\_DATEおよびJOB\_ID列には、NOT NULL制約が適用されます。

**注意:** 主キー制約の詳細は、この章で後述します。

# UNIQUE制約



ORACLE

Copyright © 2007, Oracle. All rights reserved.

## UNIQUE制約

UNIQUEキー整合性制約では、列または列セット(キー)のすべての値が一意であること、つまり表の任意の2つの行で、指定した列または列セットの値が重複することは許可されません。UNIQUEキー制約の定義に含まれる列(または列セット)は、一意キーと呼ばれます。UNIQUE制約が複数の列で構成される場合、その列のグループは複合一意キーと呼ばれます。

UNIQUE制約では、同じ列に対してNOT NULL制約も定義しないかぎり、NULLの入力が可能です。事実上、任意の数の行で、NOT NULL制約のない列の値をNULLにすることができます。これは、NULLがどの値とも等しくないと思なされるためです。列(または複合UNIQUEキーのすべての列)のNULLは、常にUNIQUE制約を満たします。

**注意:** 複数の列でのUNIQUE制約に対する検索メカニズムでは、一部がNULLである複合UNIQUEキー制約で、NULLでない列に同じ値を入力することはできません。

# UNIQUE制約

表レベルまたは列レベルで定義:

```
CREATE TABLE employees (  
    employee_id      NUMBER(6) ,  
    last_name        VARCHAR2(25) NOT NULL ,  
    email            VARCHAR2(25) ,  
    salary           NUMBER(8,2) ,  
    commission_pct   NUMBER(2,2) ,  
    hire_date        DATE NOT NULL ,  
    ...  
    CONSTRAINT emp_email_uk UNIQUE(email));
```

ORACLE

Copyright © 2007, Oracle. All rights reserved.

## UNIQUE制約(続き)

UNIQUE制約は、列レベルまたは表レベルで定義できます。複合一意キーを作成する場合は、制約を表レベルで定義します。複合キーは、単一の属性では行を一意に識別できない場合に定義します。その場合は、複数の列で構成される一意のキーを作成できます。このキーの結合された値は常に一意となり、行の識別が可能になります。

このスライドの例は、EMPLOYEES表のEMAIL列に対するUNIQUE制約に適用されます。制約の名前はEMP\_EMAIL\_UKです。

**注意:** Oracleサーバーでは、一意キーの列に対して、1つの一意の索引を暗黙的に作成することで、UNIQUE制約が適用されます。

# PRIMARY KEY制約

DEPARTMENTS PRIMARY KEY

DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
1	10 Administration	200	1700
2	20 Marketing	201	1800
3	50 Shipping	124	1500
4	60 IT	103	1400
5	80 Sales	149	2500
6	90 Executive	100	1700
7	110 Accounting	205	1700
8	190 Contracting	(null)	1700

許可されない  
(NULL値)

INSERT INTO

Public Accounting	124	2500
50 Finance	124	1500

許可されない  
(すでに50が存在)

ORACLE

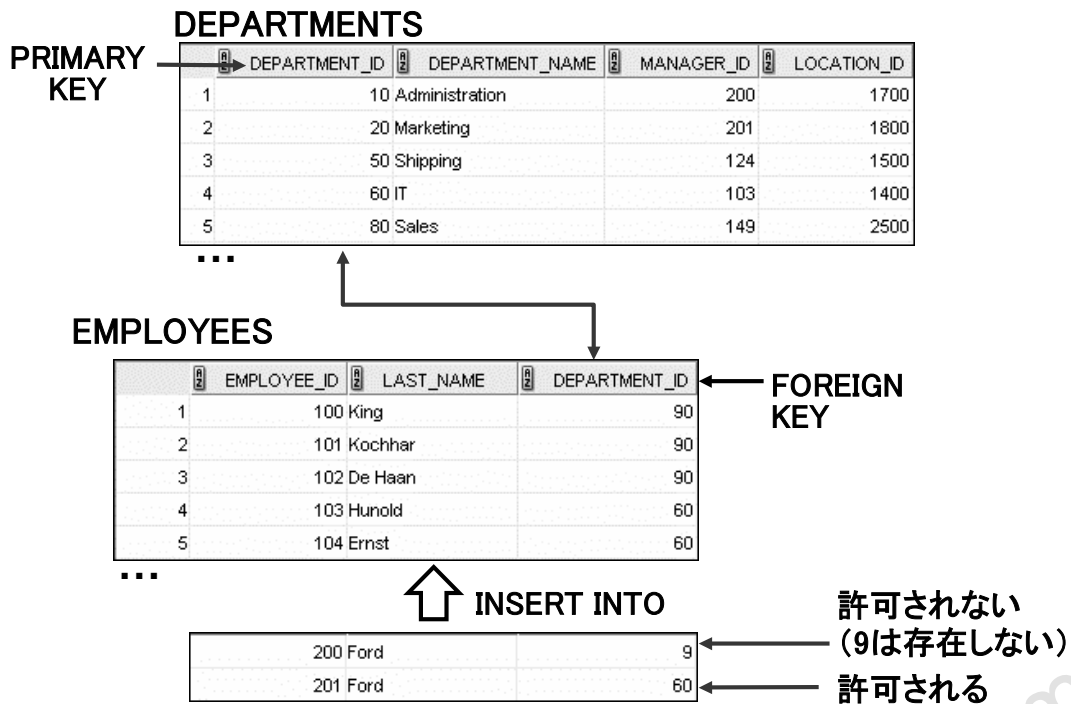
Copyright © 2007, Oracle. All rights reserved.

## PRIMARY KEY制約

PRIMARY KEY制約では、表に対する主キーが作成されます。主キーは、各表に1つだけ作成できます。PRIMARY KEY制約は、表の各行を一意に識別する列または列セットです。この制約により、その列または列の組合せが一意であることが強制され、主キーの一部であるどの列でもNULL値が許可されないことが保証されます。

**注意:** 一意であることは主キー制約の定義に含まれるため、Oracleサーバーは、主キー列に一意の索引を暗黙的に作成することで、一意であることを強制します。

# FOREIGN KEY制約



ORACLE

Copyright © 2007, Oracle. All rights reserved.

## FOREIGN KEY制約

FOREIGN KEY (または参照整合性) 制約は、列また列の組合せを外部キーとして指定し、同じ表または異なる表の主キーまたは一意キーとのリレーションシップを確立します。

このスライドの例では、EMPLOYEES表 (依存表または子表) のDEPARTMENT\_IDを外部キーとして定義しています。これは、DEPARTMENTS表 (参照先の表または親表) のDEPARTMENT\_ID列を参照します。

### ガイドライン

- 外部キーの値は、親表にある既存の値に一致するか、NULLでなければなりません。
- 外部キーは、データの値に基づいた、物理的でなく純粋に論理的なポインタです。



# FOREIGN KEY制約

表レベルまたは列レベルで定義:

```
CREATE TABLE employees (  
    employee_id      NUMBER(6),  
    last_name        VARCHAR2(25) NOT NULL,  
    email            VARCHAR2(25),  
    salary           NUMBER(8,2),  
    commission_pct   NUMBER(2,2),  
    hire_date        DATE NOT NULL,  
    ...  
    department_id    NUMBER(4),  
    CONSTRAINT emp_dept_fk FOREIGN KEY (department_id)  
        REFERENCES departments(department_id),  
    CONSTRAINT emp_email_uk UNIQUE(email));
```

ORACLE

Copyright © 2007, Oracle. All rights reserved.

## FOREIGN KEY制約(続き)

FOREIGN KEY制約は、列または表の制約レベルで定義できます。複合外部キーは、表レベルの定義を使用して作成する必要があります。

このスライドの例は、EMPLOYEES表のDEPARTMENT\_ID列で、表レベルの構文を使用してFOREIGN KEY制約を定義しています。制約の名前はEMP\_DEPT\_FKです。

制約が1つの列に基づく場合、外部キーは列レベルでも定義できます。構文は、キーワードFOREIGN KEYを使用しない点が異なります。たとえば次のようになります。

```
CREATE TABLE employees  
(  
    ...  
    department_id NUMBER(4) CONSTRAINT emp_deptid_fk  
        REFERENCES departments(department_id),  
    ...  
)
```

## FOREIGN KEY制約: キーワード

- **FOREIGN KEY:** 表制約レベルにある子表の列を定義
- **REFERENCES:** 親表にある表および列を識別
- **ON DELETE CASCADE:** 親表内の行が削除された場合、子表内の依存行を削除
- **ON DELETE SET NULL:** 依存する外部キーの値をNULLに変換

ORACLE

Copyright © 2007, Oracle. All rights reserved.

### FOREIGN KEY制約: キーワード

外部キーは子表で定義し、参照先の列を含む表を親表とします。外部キーは、次のキーワードを組み合わせて定義します。

- **FOREIGN KEY**は、表制約レベルにある子表の列を定義するために使用します。
- **REFERENCES:** 親表にある表および列を識別します。
- **ON DELETE CASCADE**は、親表内の行が削除されると、子表内の依存行も削除されることを示します。
- **ON DELETE SET NULL**は、親表内の行が削除されると、外部キーの値がNULLに設定されることを示します。

デフォルトの動作は制限ルールと呼ばれ、参照データの更新または削除を禁止します。

**ON DELETE CASCADE**または**ON DELETE SET NULL**オプションを指定しない場合は、親表で、子表が参照している行を削除できません。

## CHECK制約

- 各行が満たす必要がある条件を定義します。
- 次の式は許可されません。
  - CURRVAL、NEXTVAL、LEVELおよびROWNUM疑似列に対する参照
  - SYSDATE、UID、USERおよびUSERENV関数に対するコール
  - 他の行の他の値を参照する問合せ

```
..., salary NUMBER(2)
CONSTRAINT emp_salary_min
CHECK (salary > 0),...
```

ORACLE

Copyright © 2007, Oracle. All rights reserved.

### CHECK制約

CHECK制約は、各行が満たす必要がある条件を定義します。この条件では、問合せの条件と同じ構造体を使用できますが、次の処理は例外です。

- CURRVAL、NEXTVAL、LEVELおよびROWNUM疑似列に対する参照
- SYSDATE、UID、USERおよびUSERENV関数に対するコール
- 他の行の他の値を参照する問合せ

1つの列に対しては、制約の定義内でその列を参照するCHECK制約を複数設定できます。1つの列に定義できるCHECK制約の数に制限はありません。

CHECK制約は、列レベルまたは表レベルで定義できます。

```
CREATE TABLE employees
(
    salary NUMBER(8,2) CONSTRAINT emp_salary_min
    CHECK (salary > 0),
    ...
)
```

## CREATE TABLE: 例

```
CREATE TABLE employees
( employee_id      NUMBER(6)
  CONSTRAINT emp_employee_id PRIMARY KEY
, first_name       VARCHAR2(20)
, last_name        VARCHAR2(25)
  CONSTRAINT emp_last_name_nn NOT NULL
, email            VARCHAR2(25)
  CONSTRAINT emp_email_nn    NOT NULL
  CONSTRAINT emp_email_uk    UNIQUE
, phone_number     VARCHAR2(20)
, hire_date        DATE
  CONSTRAINT emp_hire_date_nn NOT NULL
, job_id           VARCHAR2(10)
  CONSTRAINT emp_job_nn      NOT NULL
, salary           NUMBER(8,2)
  CONSTRAINT emp_salary_ck   CHECK (salary>0)
, commission_pct   NUMBER(2,2)
, manager_id       NUMBER(6)
  CONSTRAINT emp_manager_fk REFERENCES
    employees (employee_id)
, department_id    NUMBER(4)
  CONSTRAINT emp_dept_fk    REFERENCES
    departments (department_id));
```

ORACLE

Copyright © 2007, Oracle. All rights reserved.

### CREATE TABLE: 例

このスライドの例は、HRスキーマでEMPLOYEES表を作成するための文を示しています。

## 制約違反

```
UPDATE employees
SET   department_id = 55
WHERE department_id = 110;
```

```
Error starting at line 1 in command:
UPDATE employees
SET   department_id = 55
WHERE department_id = 110
Error report:
SQL Error: ORA-02291: integrity constraint (ORA16.EMP_DEPT_FK) violated - parent key not found
02291. 00000 - "integrity constraint (%s.%s) violated - parent key not found"
*Cause:      A foreign key value has no matching primary key value.
*Action:     Delete the foreign key or add a matching primary key.
```

部門55は存在しません。

ORACLE

Copyright © 2007, Oracle. All rights reserved.

### 制約違反

列に制約を設定すると、制約ルールに違反する操作が試みられるとエラーが戻されます。たとえば、整合性制約に関連付けられている値を含むレコードを更新しようとする、エラーが戻されます。

スライドの例では、親表のDEPARTMENTSに部門55が存在しないため、「親キーがありません」というメッセージ(違反ORA-02291)が戻されます。

## 制約違反

別の表で外部キーとして使用されている主キーを含む行は、削除できません。

```
DELETE FROM departments
WHERE department_id = 60;
```

Error starting at line 1 in command: DELETE FROM departments WHERE department_id = 60 Error report: SQL Error: ORA-02292: integrity constraint (ORA16.EMP_DEPT_FK) violated - child record found 02292. 00000 - "integrity constraint (%s.%s) violated - child record found" *Cause: attempted to delete a parent key value that had a foreign dependency. *Action: delete dependencies first then parent or disable constraint.	
--	--

ORACLE

Copyright © 2007, Oracle. All rights reserved.

### 制約違反(続き)

整合性制約に関連付けられている値を含むレコードを削除しようとすると、エラーが戻されます。このスライドの例では、DEPARTMENTS表から部門60を削除しようとしませんが、その部門番号がEMPLOYEES表で外部キーとして使用されているため、エラーが戻されます。削除しようとした親レコードに子レコードがある場合は、「子レコードがあります」というメッセージ(違反ORA-02292)が戻されます。

次の文は、部門70に従業員がいないため正常に実行されます。

```
DELETE FROM departments
WHERE department_id = 70;
```

```
1 rows deleted
```

## 章の講義項目

- データベース・オブジェクト
  - ネーミング規則
- CREATE TABLE文:
  - 別のユーザーの表にアクセス
  - DEFAULTオプション
- データ型
- 制約の概要: NOT NULL、PRIMARY KEY、FOREIGN KEY、CHECK制約
- 副問合せを使用した表の作成
- ALTER TABLE
  - 読取り専用の表
- DROP TABLE文

ORACLE

Copyright © 2007, Oracle. All rights reserved.

## 副問合せを使用した表の作成

- CREATE TABLE文とAS *subquery*オプションを組み合わせて、表を作成し、行を挿入します。

```
CREATE TABLE table  
      [(column, column...)]  
AS subquery;
```

- 指定した列の番号と副問合せ列の番号を一致させます。
- 列名とデフォルト値を指定して列を定義します。

ORACLE

Copyright © 2007, Oracle. All rights reserved.

### 副問合せを使用した表の作成

表を作成する2つ目の方法は、AS *subquery*句を適用することです。これにより表が作成されるとともに、副問合せで戻された行が挿入されます。

構文の内容

table	表の名前
column	列の名前、デフォルト値および整合性制約
subquery	新しい表に挿入する行のセットを定義するSELECT文

ガイドライン

- 表は指定した列名で作成され、SELECT文で取得された行が表に挿入されます。
- 列の定義に含めることができるのは、列名とデフォルト値だけです。
- 列の仕様が指定されている場合、その列数は副問合せSELECTリスト内の列数に一致する必要があります。
- 列の仕様が指定されていない場合、表の列名は、副問合せ内の列名と同じになります。
- 列のデータ型の定義とNOT NULL制約は、新しい表に渡されます。ただし、継承されるのは明示的なNOT NULL制約だけです。PRIMARY KEY列は新しい列にNOT NULL機能を渡しません。その他の制約ルールは新しい表に渡されません。ただし、列の定義に制約を追加することができます。



## 副問合せを使用した表の作成

```
CREATE TABLE dept80
AS
  SELECT employee_id, last_name,
         salary*12 ANNSAL,
         hire_date
  FROM   employees
 WHERE  department_id = 80;
```

CREATE TABLE succeeded.

```
DESCRIBE dept80
```

Name	Null	Type
EMPLOYEE_ID		NUMBER(6)
LAST_NAME	NOT NULL	VARCHAR2(25)
ANNSAL		NUMBER
HIRE_DATE	NOT NULL	DATE

ORACLE

Copyright © 2007, Oracle. All rights reserved.

### 副問合せを使用した表の作成(続き)

スライドの例では、DEPT80という名前の表を作成し、部門80に勤務するすべての従業員の詳細データを含めます。DEPT80表のデータがEMPLOYEES表から取得されていることに注意してください。

DESCRIBEコマンドを使用することで、データベース表の存在を確認し、列の定義をチェックできます。

ただし、式を選択する場合は列に別名を付けてください。式SALARY\*12には別名ANNSALが付けられています。別名がないと、次のエラーが生成されます。

```
Error starting at line 1 in command:
CREATE TABLE   dept80
AS SELECT  employee_id, last_name,
salary*12,
hire_date FROM employees WHERE   department_id = 80
Error at Command Line:3 Column:6

Error report:
SQL Error: ORA-00998: must name this expression with a column alias
00998. 00000 - "must name this expression with a column alias"
*Cause:
*Action:
```

## 章の講義項目

- データベース・オブジェクト
  - ネーミング規則
- CREATE TABLE文:
  - 別のユーザーの表にアクセス
  - DEFAULTオプション
- データ型
- 制約の概要: NOT NULL、PRIMARY KEY、FOREIGN KEY、CHECK制約
- 副問合せを使用した表の作成
- ALTER TABLE
  - 読取り専用の表
- DROP TABLE文

ORACLE

Copyright © 2007, Oracle. All rights reserved.

# ALTER TABLE文

ALTER TABLE文は次の場合に使用します。

- 新しい列の追加
- 既存の列定義の変更
- 新しい列のデフォルト値の定義
- 列の削除
- 列の名前の変更
- 読取り専用状態への表の変更

ORACLE

Copyright © 2007, Oracle. All rights reserved.

## ALTER TABLE文

表の作成後に、次のいずれかの理由で、表構造の変更が必要になる場合があります。

- 列を省略したため
- 列の定義または名前を変更する必要があるため
- 列を削除する必要があるため
- 表を読取り専用モードにするため

これはALTER TABLE文を使用して実行できます。

## 読取り専用の表

表を読取り専用モードにするには、ALTER TABLE構文を使用します。

- 表のメンテナンス中にDDLまたはDMLによる変更を防ぎます。
- 読取り/書込みモードに戻します。

```
ALTER TABLE employees READ ONLY;

-- perform table maintenance and then
-- return table back to read/write mode

ALTER TABLE employees READ WRITE;
```

ORACLE

Copyright © 2007, Oracle. All rights reserved.

### 読取り専用の表

Oracle Database 11gでは、READ ONLYを指定して表を読取り専用モードにすることができます。表がREAD-ONLYモードである場合は、その表に影響するDML文またはSELECT ... FOR UPDATE文を発行できません。DDL文は、表内のデータを変更しないものであれば発行できます。表に関連付けられている索引に対する操作は、表がREAD ONLYモードである場合のみ許可されます。

読取り専用の表を読取り/書込みモードに戻すには、READ/WRITEを指定します。

**注意:** READ ONLYモードである表は削除できます。DROPコマンドは、データ・ディクショナリ内でのみ実行されるため、表のコンテンツに対するアクセスは不要です。表で使用された領域は、表領域が再び読取り/書込みになるまで再利用されず、その後、ブロック・セグメント・ヘッダーなどに必要な変更を加えることができます。

ALTER TABLE文の詳細は、『Oracle Database 11g: 入門 SQL 基礎 II Ed 1』コースを参照してください。

## 章の講義項目

- データベース・オブジェクト
  - ネーミング規則
- CREATE TABLE文:
  - 別のユーザーの表にアクセス
  - DEFAULTオプション
- データ型
- 制約の概要: NOT NULL、PRIMARY KEY、FOREIGN KEY、CHECK制約
- 副問合せを使用した表の作成
- ALTER TABLE
  - 読取り専用の表
- DROP TABLE文

ORACLE

Copyright © 2007, Oracle. All rights reserved.

## 表の削除

- 表がごみ箱に移動されます。
- PURGE句を指定すると表およびすべてのデータが完全に削除されます。
- 依存オブジェクトが無効になり、表に対するオブジェクト権限が削除されます。

```
DROP TABLE dept80;
```

```
DROP TABLE dept80 succeeded.
```

ORACLE

Copyright © 2007, Oracle. All rights reserved.

### 表の削除

DROP TABLE文を使用すると、表がごみ箱に移動されるか、または表とそのすべてのデータがデータベースから完全に削除されます。PURGE句を指定しないかぎり、DROP TABLE文を実行しても、他のオブジェクトが使用できるように領域が解放されて表領域に戻されることはなく、その領域はユーザーの領域割当てとして計算されたままになります。表を削除することで、依存オブジェクトが無効になり、表に対するオブジェクト権限が削除されます。

表を削除すると、その表のすべてのデータと、関連付けられているすべての索引がデータベースから失われます。

#### 構文

```
DROP TABLE table [PURGE]
```

この構文で、*table*は表の名前です。

#### ガイドライン

- すべてのデータが表から削除されます。
- ビューおよびシノニムは残りますが、無効になります。
- 保留中のトランザクションはコミットされます。
- 表を削除できるのは、表の作成者またはDROP ANY TABLE権限を持つユーザーだけです。

**注意:** 削除された表をごみ箱からリストアするには、FLASHBACK TABLE文を使用します。詳細は、『Oracle Database 11g: 入門 SQL 基礎 II Ed 1』コースで説明しています。

## まとめ

この章では、CREATE TABLE文を使用して表を作成し、制約を設定する方法を学習しました。

- 主要なデータベース・オブジェクトの分類
- 表構造の説明
- 列で使用可能なデータ型のリスト
- 単純な表の作成
- 表の作成時における制約の作成方法の説明
- スキーマ・オブジェクトの機能の説明

ORACLE

Copyright © 2007, Oracle. All rights reserved.

## まとめ

この章では、次の文の使用方法について学習しました。

### CREATE TABLE

- CREATE TABLE文は、表の作成および制約の設定に使用します。
- 副問合せを使用して、別の表に基づいて表を作成します。

### DROP TABLE

- 行と表構造を削除します。
- 実行すると、この文はロールバックできません。

## 演習10: 概要

この演習では次の項目について説明しています。

- 新しい表の作成
- CREATE TABLE AS構文を使用した新しい表の作成
- 表の存在の確認
- 読取り専用状態への表の設定
- 表の削除

ORACLE

Copyright © 2007, Oracle. All rights reserved.

### 演習10: 概要

CREATE TABLE文を使用して新しい表を作成します。新しい表がデータベースに追加されたことを確認します。また、表の状態をREAD ONLYとして設定し、READ/WRITEに戻す方法も学習します。

**注意:** すべてのDDLおよびDML文について、「Run Script」アイコンをクリックして(または[F5]を押して)SQL Developerで問合せを実行します。これにより、「Script Output」タブ・ページにフィードバック・メッセージが表示されます。SELECT問合せの場合は、さらに「Execute Statement」アイコンをクリックするか[F9]を押すと、「Results」タブ・ページに書式設定された出力が表示されます。



## 演習10

1. 次の表インスタンスのチャートに基づいてDEPT表を作成します。lab\_10\_01.sqlというスクリプトに文を保存し、スクリプト内の文を実行して表を作成します。表が作成されたことを確認してください。

列名	ID	NAME
キーの種類	Primary key	
NULL/一意		
FK表		
FK列		
データ型	NUMBER	VARCHAR2
長さ	7	25

Name	Null	Type
-----	-----	-----
ID	NOT NULL	NUMBER(7)
NAME		VARCHAR2(25)

2. DEPT表にDEPARTMENTS表からデータを移入します。必要な列だけを含めます。
3. 次の表インスタンスのチャートに基づいてEMP表を作成します。lab\_10\_03.sqlというスクリプトに文を保存し、スクリプト内の文を実行して表を作成します。表が作成されたことを確認してください。

列名	ID	LAST_NAME	FIRST_NAME	DEPT_ID
キーの種類				
NULL/一意				
FK表				DEPT
FK列				ID
データ型	NUMBER	VARCHAR2	VARCHAR2	NUMBER
長さ	7	25	25	7

Name	Null	Type
-----	-----	-----
ID		NUMBER(7)
LAST_NAME		VARCHAR2(25)
FIRST_NAME		VARCHAR2(25)
DEPT_ID		NUMBER(7)

## 演習10(続き)

- EMPLOYEES表の構造に基づいてEMPLOYEES2表を作成します。EMPLOYEE\_ID、FIRST\_NAME、LAST\_NAME、SALARYおよびDEPARTMENT\_ID列のみを含めます。新しい表の列名をそれぞれID、FIRST\_NAME、LAST\_NAME、SALARYおよびDEPT\_IDとします。
- EMPLOYEES2表の状態を読取り専用に変更します。
- 次の行をEMPLOYEES2表に挿入してみます。

ID	FIRST_NAME	LAST_NAME	SALARY	DEPT_ID
34	Grant	Marcie	5678	10

次のエラー・メッセージが表示されます。

```
Error starting at line 1 in command:
INSERT INTO employees2
VALUES (34, 'Grant','Marcie',5678,10)
Error at Command Line:1 Column:12
Error report:
SQL Error: ORA-12081: update operation not allowed on table "ORA16"."EMPLOYEES2"
12081. 00000 - "update operation not allowed on table \"%s\".\"%s\""
*Cause:      An attempt was made to update a read-only materialized view.
*Action:     No action required. Only Oracle is allowed to update a
              read-only materialized view.
```

- EMPLOYEES2表を読取り/書込み状態に戻します。ここで、再び同じ行の挿入を試みます。次のメッセージが表示されます。

```
ALTER TABLE employees2 succeeded.
1 rows inserted
```

- EMPLOYEES2表を削除します。

# 11

## その他のスキーマ・オブジェクトの作成

ORACLE

Copyright © 2007, Oracle. All rights reserved.

Oracle Internal & Oracle Academy  
Use Only

## 目的

この章を終えると、次のことができるようになります。

- 単一ビューおよび複合ビューの作成
- ビューのデータの取得
- 順序の作成、メンテナンスおよび使用
- 索引の作成およびメンテナンス
- プライベートおよびパブリック・シノニムの作成

ORACLE

Copyright © 2007, Oracle. All rights reserved.

### 目的

この章では、ビュー、順序、シノニムおよび索引オブジェクトについて説明します。ビュー、順序および索引を作成して使用する基本的な方法について学習します。

## 章の講義項目

- ビューの概要:
  - ビューのデータの作成、変更および取得
  - ビューでのデータ操作言語(DML)の操作
  - ビューの削除
- 順序の概要:
  - 順序の作成、使用および変更
  - 順序値のキャッシュ
  - NEXTVALおよびCURRVAL疑似列
- 索引の概要
  - 索引の作成および削除
- シノニムの概要
  - シノニムの作成および削除

ORACLE

Copyright © 2007, Oracle. All rights reserved.

# データベース・オブジェクト

オブジェクト	説明
表	行で構成される、記憶域の基本単位。
ビュー	1つ以上の表のデータのサブセットを論理的に表します。
順序	数値を生成します。
索引	データを取得する問合せのパフォーマンスを向上させます。
シノニム	オブジェクトに別名を付けます。

ORACLE

Copyright © 2007, Oracle. All rights reserved.

## データベース・オブジェクト

データベースには、表の他にいくつかのオブジェクトがあります。この章では、ビュー、順序、索引およびシノニムについて学習します。

ビューでは、表のデータを表示または非表示にできます。

多くのアプリケーションでは、主キーの値として一意の数字を使用する必要があります。それには、この要件に対処するためのコードをアプリケーションに組み込む方法、または順序を使用して一意の数字を生成する方法があります。

データ取得の問合せのパフォーマンスを向上させる場合は、索引の作成を検討してください。

索引は、列または列コレクションに対して一意性を実現する場合にも使用できます。

シノニムを使用すると、オブジェクトに別名を付けることができます。

# ビューの概要

EMPLOYEES表

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY
100	Steven	King	SKING	515.123.4567	17-JUN-87	AD_PRES	24000
101	Neena	Kochhar	NKOCHH...	515.123.4568	21-SEP-89	AD_VP	17000
102	Lex	De Haan	LDEHAAN	515.123.4569	13-JAN-93	AD_VP	17000
103	Alexander	Hunold	AHUNOLD	590.423.4567	03-JAN-90	IT_PROG	9000
104	Bruce	Ernst	BERNST	590.423.4568	09-SEP-98	IT_PROG	6000
105	David	Turner	DTURNER	515.123.4569	07-SEP-97	SA_REP	11000
106	Alex	Baer	ABEAR	515.123.4567	17-MAR-97	SA_REP	8600
107	Julia	Abel	JABEL	515.123.4567	17-APR-97	SA_REP	7000
108	Keith	Johnson	KJHNSN	515.123.4567	17-SEP-87	AD_ASST	4400
109	Glen	Smith	GSMITH	515.123.4567	17-FEB-96	MK_MAN	13000
110	Greg	Clayton	GCLAYTON	515.123.4567	17-AUG-97	MK_REP	6000
111	Timothy	Gietz	TGIEZT	515.123.4567	15-SEP-98	IT_PROG	6000
112	Lisa	Green	LGREEN	515.123.4567	17-MAR-97	SA_REP	8000
113	Mark	Winters	MWINTERS	515.123.4567	17-DEC-97	SA_REP	9500
114	Janice	King	JKING	515.123.4567	17-MAR-97	SA_REP	8000
115	Shelley	Higgins	SHIGGINS	515.123.8080	07-JUN-94	AC_MGR	12000
116	William	Gietz	WGIEZT	515.123.8181	07-JUN-94	AC_ACC...	8300

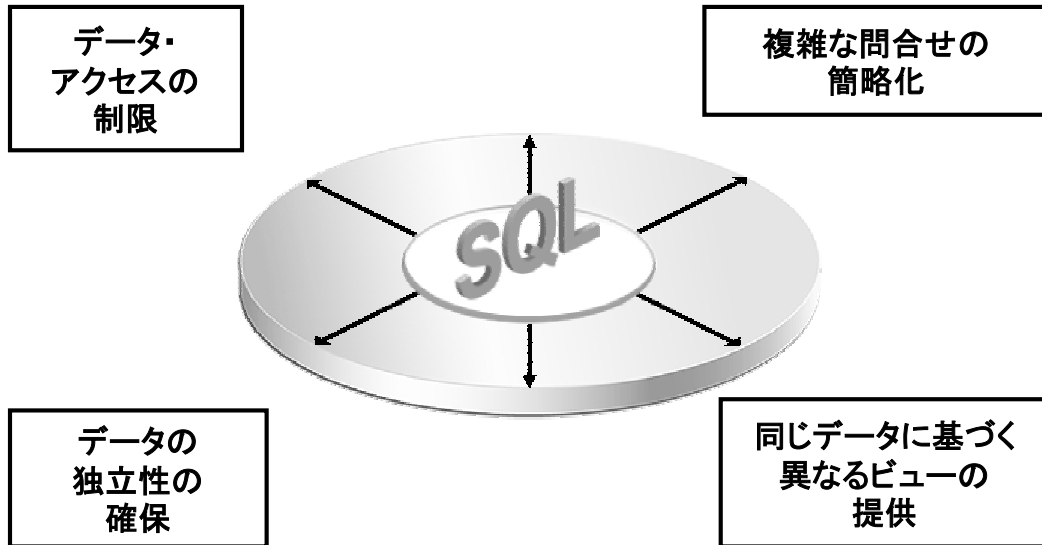
ORACLE

Copyright © 2007, Oracle. All rights reserved.

## ビューの概要

表のビューを作成すると、データの論理的なサブセットまたは組合せを表示できます。ビューは、表または別のビューに基づく論理的な表です。ビューは、独自にデータを持つことなく、表のデータを表示または変更できるウィンドウのように機能します。ビューの基になる表は実表と呼ばれます。ビューは、SELECT文としてデータ・ディクショナリに格納されます。

## ビューの利点



ORACLE

Copyright © 2007, Oracle. All rights reserved.

### ビューの利点

- ビューには表から選択した列が表示されるため、データへのアクセスが制限されます。
- ビューを使用すると、単純な問合せによって複雑な問合せの結果を取得できます。たとえば、ユーザーが結合文の記述方法を知らなくても、ビューを使用することで、複数の表に対して情報の問合せを実行できます。
- ビューにより、臨時のユーザーおよびアプリケーション・プログラムに対してデータの独立性が提供されます。1つのビューで、複数の表からデータを取得できます。
- ビューにより、ユーザーのグループが特定の基準に従ってデータにアクセスできます。

詳細は、『Oracle Database SQL言語リファレンス11gリリース1(11.1)』のCREATE VIEWに関する節を参照してください。



## 単一ビューと複合ビュー

機能	単一ビュー	複合ビュー
表の数	1つ	1つ以上
関数を含む	いいえ	はい
データのグループを含む	いいえ	はい
ビューを通じたDML操作	はい	必ずしも可能ではない

ORACLE

Copyright © 2007, Oracle. All rights reserved.

### 単一ビューと複合ビュー

ビューは、単一ビューと複合ビューの2つに分類できます。基本的な違いは、DML (INSERT、UPDATEおよびDELETE) の操作にあります。

- 単一ビューの特徴
  - 1つの表のみからデータを導入する
  - 関数またはデータのグループを含まない
  - ビューを通じてDML操作を実行可能
- 複合ビューの特徴
  - 複数の表からデータを導出する
  - 関数またはデータのグループを含む
  - ビューを通じたDML操作が許可されない場合がある

## ビューの作成

- CREATE VIEW文に副問合せを埋め込みます。

```
CREATE [OR REPLACE] [FORCE|NOFORCE] VIEW view
[(alias[, alias]...)]
AS subquery
[WITH CHECK OPTION [CONSTRAINT constraint]]
[WITH READ ONLY [CONSTRAINT constraint]];
```

- 副問合せに、複雑なSELECT構文を含めることができます。

ORACLE

Copyright © 2007, Oracle. All rights reserved.

### ビューの作成

CREATE VIEW文に副問合せを埋め込んで、ビューを作成することができます。

構文の内容

OR REPLACE

ビューがすでに存在する場合は再作成されます。

FORCE

実表の有無にかかわらずビューが作成されます。

NOFORCE

実表がある場合のみビューが作成されます(デフォルト)。

*view*

ビューの名前です。

*alias*

ビューの問合せによって選択された式の名前を指定します。  
(別名のはビューによって選択された式の数に一致する必要があります)

*subquery*

完全なSELECT文(SELECTリストの列には別名を使用できる)

WITH CHECK OPTION

ビューにアクセス可能な行だけを挿入または更新できることを指定します。

*constraint*

CHECK OPTION制約に割り当てられる名前です。

WITH READ ONLY

このビューでDML操作を実行できないようにします。

**注意:** SQL Developerで、「Run Script」アイコンをクリックするか、[F5]を押してデータ定義言語(DDL)文を実行します。「Script Output」タブ・ページにフィードバック・メッセージが表示されます。

## ビューの作成

- 部門80の従業員の詳細が表示されるEMPVU80ビューを作成します。

```
CREATE VIEW empvu80
AS SELECT employee_id, last_name, salary
FROM employees
WHERE department_id = 80;
CREATE VIEW succeeded.
```

- SQL\*Plus DESCRIBEコマンドを使用して、ビューの構造を記述します。

```
DESCRIBE empvu80
```

ORACLE

Copyright © 2007, Oracle. All rights reserved.

### ビューの作成(続き)

このスライドの例では、部門80の各従業員の従業員番号、姓および給与が表示されるビューを作成します。

DESCRIBEコマンドを使用すると、ビューの構造を表示できます。

Name	Null	Type
EMPLOYEE_ID	NOT NULL	NUMBER(6)
LAST_NAME	NOT NULL	VARCHAR2(25)
SALARY		NUMBER(8,2)

### ガイドライン

- ビューを定義する副問合せには、結合、グループ、副問合せなど、複雑なSELECT構文を含めることができます。
- WITH CHECK OPTIONで作成したビューに対して制約の名前を指定しない場合は、デフォルトの名前がSYS\_Cnの形式で割り当てられます。
- OR REPLACEオプションを使用すると、ビューの定義を削除および再作成することなく、またその定義に付与されていたオブジェクト権限を再度付与することなく、ビューの定義を変更できます。

## ビューの作成

- 副問合せ内で列の別名を使用して、ビューを作成します。

```
CREATE VIEW   salvu50
  AS SELECT   employee_id ID_NUMBER, last_name NAME,
              salary*12 ANN_SALARY
  FROM        employees
  WHERE       department_id = 50;

CREATE VIEW succeeded.
```

- このビューから列を選択するには、指定した別名を使用します。

ORACLE

Copyright © 2007, Oracle. All rights reserved.

### ビューの作成(続き)

列の名前は、副問合せに列の別名を含めることで制御できます。

このスライドの例では、部門50の全従業員の従業員番号(EMPLOYEE\_ID)と別名のID\_NUMBER、名前(LAST\_NAME)と別名のNAME、年間所得(SALARY)と別名のANN\_SALARYが含まれるビューを作成します。

別名を、CREATE文とSELECT副問合せの間に指定する方法もあります。リスト内に指定する別名の数、副問合せで選択される式の数に一致する必要があります。

```
CREATE OR REPLACE VIEW salvu50 (ID_NUMBER, NAME, ANN_SALARY)
  AS SELECT employee_id, last_name, salary*12
  FROM      employees
  WHERE     department_id = 50;
```

```
CREATE VIEW succeeded.
```

## ビューからのデータの取得

```
SELECT *  
FROM salvu50;
```

	ID_NUMBER	NAME	ANN_SALARY
1	124	Mourgos	69600
2	141	Rajs	42000
3	142	Davies	37200
4	143	Matos	31200
5	144	Vargas	30000

ORACLE

Copyright © 2007, Oracle. All rights reserved.

### ビューからのデータの取得

ビューのデータは、表と同様に取得できます。ビュー全体、または特定の行および列のみの内容を表示できます。

## ビューの変更

- EMPVU80ビューをCREATE OR REPLACE VIEW句を使用して変更します。それぞれの列の名前の別名を追加します。

```
CREATE OR REPLACE VIEW empvu80
(id_number, name, sal, department_id)
AS SELECT  employee_id, first_name || ' '
           || last_name, salary, department_id
FROM      employees
WHERE     department_id = 80;
CREATE OR REPLACE VIEW succeeded.
```

- CREATE OR REPLACE VIEW句の列の別名は、副問合せの列と同じ順序でリストされています。

ORACLE

Copyright © 2007, Oracle. All rights reserved.

### ビューの変更

OR REPLACEオプションを使用すると、すでに同じ名前を持つビューがある場合でもビューを作成できます。この操作により、その所有者の古いバージョンのビューが置き換えられます。つまり、オブジェクト権限を削除、再作成および再付与することなくビューを変更できることを意味します。

**注意:** CREATE OR REPLACE VIEW句で列の別名を割り当てる場合は、副問合せの列と同じ順序で別名をリストするようにしてください。

## 複合ビューの作成

2つの表の値を表示するグループ関数を含む、複合ビューを作成します。

```
CREATE OR REPLACE VIEW dept_sum_vu
(name, minsal, maxsal, avgsal)
AS SELECT    d.department_name, MIN(e.salary),
            MAX(e.salary), AVG(e.salary)
FROM        employees e JOIN departments d
ON          (e.department_id = d.department_id)
GROUP BY d.department_name;
CREATE OR REPLACE VIEW succeeded.
```

ORACLE

Copyright © 2007, Oracle. All rights reserved.

### 複合ビューの作成


このスライドの例では、部門ごとに部門名、給与の最低額、最高額および平均が表示される、複合ビューを作成します。このビューでは別名が指定されています。別名の使用は、ビューのいずれかの列を関数または式から導出する場合に必要になります。

DESCRIBEコマンドを使用すると、ビューの構造を表示できます。ビューの内容を表示するには、SELECT文を実行します。

```
SELECT *
FROM    dept_sum_vu;
```

	NAME	MINSAL	MAXSAL	AVGSAL
1	Administration	4400	4400	4400
2	Accounting	8300	12000	10150
3	IT	4200	9000	6400
4	Executive	17000	24000	19333.3333333333333333...
5	Shipping	2500	5800	3500
6	Sales	8600	11000	10033.3333333333333333...
7	Marketing	6000	13000	9500

## ビューでDML操作を実行するためのルール

- 通常は、単一ビューでDML操作を実行できます。✓
- ビューに次のものが含まれる場合、行を削除することはできません。
  - グループ関数
  - GROUP BY句
  - DISTINCTキーワード
  - 疑似列ROWNUMキーワード

ORACLE

Copyright © 2007, Oracle. All rights reserved.

### ビューでDML操作を実行するためのルール

ビューを通じて、特定のルールに従ったDML操作をデータに対して実行できます。DML操作が一定のルールに従っていれば、ビューを通じてデータのDML操作を実行できます。

次のどの要素も含まない場合に、ビューから行を削除できます。

- グループ関数
- GROUP BY句
- DISTINCTキーワード
- 疑似列ROWNUMキーワード



## ビューでDML操作を実行するためのルール

ビューに次のものが含まれる場合、ビューのデータを変更できません。

- グループ関数
- GROUP BY句
- DISTINCTキーワード
- 疑似列ROWNUMキーワード
- 式によって定義された列

ORACLE

Copyright © 2007, Oracle. All rights reserved.

### ビューでDML操作を実行するためのルール(続き)

前のスライドで示した条件項目、または式(SALARY \* 12など)によって定義された列がビューに含まれない場合は、ビューを通じてデータを変更できます。

## ビューでDML操作を実行するためのルール

ビューに次のものが含まれる場合、ビューを通じてデータを追加することはできません。

- グループ関数
- GROUP BY句
- DISTINCTキーワード
- 疑似列ROWNUMキーワード
- 式によって定義された列
- ビューで選択されていない、実表内のNOT NULL列

ORACLE

Copyright © 2007, Oracle. All rights reserved.

### ビューでDML操作を実行するためのルール(続き)

スライドに示した項目がビューに含まれていない場合は、ビューを通じてデータを追加できます。実表にデフォルト値がなく、ビューにNOT NULL列が含まれる場合は、ビューにデータを追加することはできません。必要なすべての値がビューに表示されなければなりません。値は、ビューを通じて基底の表に直接追加されていることに注意してください。

詳細は、『Oracle Database SQL言語リファレンス11gリリース1(11.1)』のCREATE VIEWに関する節を参照してください。

## WITH CHECK OPTION句の使用法

- WITH CHECK OPTION句を使用すると、ビューで実行したDML操作がビューのドメイン内に保持されるようになります。

```
CREATE OR REPLACE VIEW empvu20
AS SELECT      *
   FROM        employees
   WHERE       department_id = 20
   WITH CHECK OPTION CONSTRAINT empvu20 ck ;

CREATE OR REPLACE VIEW succeeded.
```

- department\_idが20以外である行をINSERTで挿入する操作、またはビュー内の行の部門番号をUPDATEで更新する操作を試行しても、WITH CHECK OPTION制約に違反するため失敗します。

ORACLE

Copyright © 2007, Oracle. All rights reserved.

### WITH CHECK OPTION句の使用法

ビューを通じて参照整合性チェックを実行できます。また、データベース・レベルで制約を適用することもできます。ビューは、データの整合性の保護を目的として使用できますが、使用範囲は非常に限定されています。

WITH CHECK OPTION句を使用すると、ビューを通じて実行されたINSERTおよびUPDATEによって、ビューが選択できない行を作成できないことが指定されます。これにより、挿入または更新されるデータに対して、整合性制約とデータ検証チェックの実行が有効になります。ビューで選択されていない行に対してDML操作が試みられると、エラーおよび制約名(指定されている場合)が表示されます。

```
UPDATE empvu20
SET    department_id = 10
WHERE  employee_id = 201;
```

次のようなメッセージが表示されます。

```
Error report:
SQL Error: ORA-01402: view WITH CHECK OPTION where-clause violation
01402. 00000 - "view WITH CHECK OPTION where-clause violation"
```

**注意:** 部門番号を10に変更すると、ビューにはその従業員が参照できなくなるため、どの行も更新されません。したがって、WITH CHECK OPTION句を使用すると、ビューには部門20の従業員のみが表示可能になり、ビューを通じてそれらの従業員の部門番号を変更することはできなくなります。

## DML操作の拒否

- ビューの定義にWITH READ ONLYオプションを追加すると、DML操作が実行されないようにできます。
- ビュー内の行でDML操作を実行しようとする、Oracleサーバーでエラーが発生します。



ORACLE

Copyright © 2007, Oracle. All rights reserved.

### DML操作の拒否

WITH READ ONLYオプションを使用してビューを作成すると、ビューでDML操作が実行されないようにすることができます。次のスライドの例では、EMPVU10ビューを変更して、ビューでDML操作を実行できないようにしています。

## DML操作の拒否

```
CREATE OR REPLACE VIEW empvu10
  (employee_number, employee_name, job_title)
AS SELECT      employee_id, last_name, job_id
  FROM        employees
  WHERE       department_id = 10
  WITH READ ONLY ;
```

CREATE OR REPLACE VIEW succeeded.

ORACLE

Copyright © 2007, Oracle. All rights reserved.

### DML操作の拒否(続き)

読取り専用制約が設定されているビューから行を削除しようとすると、エラーが発生します。

```
DELETE FROM empvu10
WHERE employee_number = 200;
```

同様に、読取り専用制約が設定されているビューを使用して、行の挿入または行の変更を試みた場合も、同じエラーが発生します。

```
Error report:
SQL Error: ORA-42399: cannot perform a DML operation on a read-only view
```

## ビューの削除

ビューはデータベース内の基礎となる表から作成されているため、データを失うことなくビューを削除できます。

```
DROP VIEW view;
```

```
DROP VIEW empvu80;
```

```
DROP VIEW empvu80 succeeded.
```

ORACLE

Copyright © 2007, Oracle. All rights reserved.

### ビューの削除

ビューを削除するにはDROP VIEW文を使用します。この文は、データベースからビューの定義を削除します。ただし、ビューを削除しても、ビューの基になっている表には影響ありません。これに対し、削除したビューに基づいていたビューまたはその他のアプリケーションは無効になります。ビューを削除できるのは、作成者またはDROP ANY VIEW権限を持つユーザーだけです。

構文の内容

*view*            ビューの名前

## 演習11: パート1の概要

この演習では次の項目について説明しています。

- 単一ビューの作成
- 複合ビューの作成
- チェック制約を設定したビューの作成
- ビュー内のデータの変更試行
- ビューの削除

ORACLE

Copyright © 2007, Oracle. All rights reserved.

### 演習11: パート1の概要

この章の演習のパート1では、ビューの作成、使用および削除に関する様々な演習問題に取り組めます。この章の最後に示す、1から6の問題を解きます。

## 章の講義項目

- ビューの概要:
  - ビューのデータの作成、変更および取得
  - ビューでのDML操作
  - ビューの削除
- 順序の概要:
  - 順序の作成、使用および変更
  - 順序値のキャッシュ
  - NEXTVALおよびCURRVAL疑似列
- 索引の概要
  - 索引の作成および削除
- シノニムの概要
  - シノニムの作成および削除

ORACLE

Copyright © 2007, Oracle. All rights reserved.



## 順序

オブジェクト	説明
表	行で構成される、記憶域の基本単位。
ビュー	1つ以上の表のデータのサブセットを論理的に表します。
順序	数値を生成します。
索引	一部の問合せのパフォーマンスが向上します。
シノニム	オブジェクトに別名を付けます。

ORACLE

Copyright © 2007, Oracle. All rights reserved.

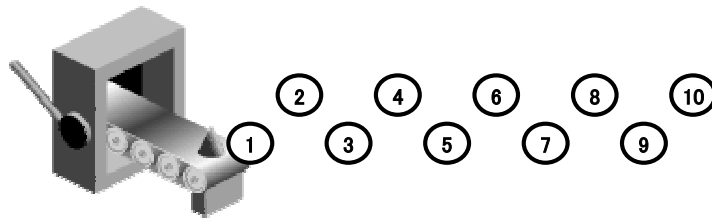
### 順序

順序は、整数の値を作成するデータベース・オブジェクトです。順序を作成した後で、その順序を使用して番号を生成することができます。

# 順序

## 順序の概要

- 一意の番号を自動的に生成できます。
- 共有可能なオブジェクトです。
- 主キーの値の作成に使用できます。
- アプリケーション・コードの代わりに使用できます。
- メモリーにキャッシュすると、順序値へのアクセス効率が向上します。



ORACLE

Copyright © 2007, Oracle. All rights reserved.

## 順序(続き)

順序は、ユーザーが作成するデータベース・オブジェクトで、複数のユーザーが共有して整数の生成に使用できます。

順序は、一意の値を生成するか、または同じ番号をリサイクルして再利用するように定義できます。

一般的に、順序は、各行で一意でなければならない主キーの値の作成に使用されます。順序は、内部のOracleルーチンによって生成され、増分または減分が行われます。このオブジェクトを使用すると、順序を生成するルーチンを記述するために必要なアプリケーション・コードの量が少なくなるため、時間の節約につながります。

順序の番号は、表から独立して格納および生成されます。そのため、同じ順序を複数の表に使用できます。

## CREATE SEQUENCE文: 構文

連続的な番号を自動的に生成する順序を定義します。

```
CREATE SEQUENCE sequence
  [INCREMENT BY n]
  [START WITH n]
  [{MAXVALUE n | NOMAXVALUE}]
  [{MINVALUE n | NOMINVALUE}]
  [{CYCLE | NOCYCLE}]
  [{CACHE n | NOCACHE}];
```

ORACLE

Copyright © 2007, Oracle. All rights reserved.

### CREATE SEQUENCE文: 構文

CREATE SEQUENCE文を使用すると、連続的な番号が自動的に生成されます。

構文の内容

*sequence*

順序の名前

INCREMENT BY *n*

順序番号の間隔を指定します。

*n*は整数です(この句を省略すると、順序は1ずつ増分される)。

START WITH *n*

最初に生成される順序番号を指定します

(この句を省略すると、順序は1から開始される)。

MAXVALUE *n*

順序で生成可能な最大値を指定します。

NOMAXVALUE

昇順の場合は最大値 $10^{27}$ 、降順の場合は最大値-1を指定します(デフォルトのオプション)。

MINVALUE *n*

順序の最小値を指定します。

NOMINVALUE

昇順の場合は最小値1、降順の場合は最小値 $-(10^{26})$ を指定します(デフォルトのオプション)。

## 順序の作成

- DEPARTMENTS表の主キーとして使用される、DEPT\_DEPTID\_SEQという順序を作成します。
- CYCLEオプションは使用しません。

```
CREATE SEQUENCE dept_deptid_seq  
    INCREMENT BY 10  
    START WITH 120  
    MAXVALUE 9999  
    NOCACHE  
    NOCYCLE;
```

```
CREATE SEQUENCE succeeded.
```

ORACLE

Copyright © 2007, Oracle. All rights reserved.

### 順序の作成(続き)

CYCLE | NOCYCLE

最大値または最小値に達した後も順序によって引き続き値が生成されるかどうかを指定します(デフォルトのオプションはNOCYCLE)。

CACHE *n* | NOCACHE

Oracleサーバーが事前に割り当て、メモリーに保持する値の数を指定します(デフォルトでは、Oracleサーバーは20個の値をキャッシュする)。

このスライドの例では、DEPARTMENTS表のDEPARTMENT\_ID列で使用される、DEPT\_DEPTID\_SEQという順序を作成します。この順序は120から開始され、キャッシュは許可されず、循環しません。

順序を主キーの値の生成に使用する場合は、順序の循環より速く古い行をページする信頼できるメカニズムがないかぎり、CYCLEオプションを使用しないでください。

詳細は、『Oracle Database SQL言語リファレンス11gリリース1(11.1)』のCREATE SEQUENCEに関する節を参照してください。

**注意:** 順序は表に関連付けられません。通常、順序には使用目的に従った名前を付けますが、順序は名前に関係なくどこでも使用できます。

## NEXTVALおよびCURRVAL疑似列

- NEXTVALは、次に使用可能な順序値を戻します。ユーザーが異なる場合でも、参照するたびに一意の値を戻します。
- CURRVALは、現行の順序値を取得します。
- NEXTVALは、CURRVALに値を取得する前に、順序に対して実行する必要があります。

ORACLE

Copyright © 2007, Oracle. All rights reserved.

### NEXTVALおよびCURRVAL疑似列

順序を作成すると、この順序によって表に使用する順序番号が生成されます。順序値は、NEXTVALおよびCURRVAL疑似列を使用して参照します。

NEXTVAL疑似列は、指定した順序から、連続する順序番号を抽出するために使用します。NEXTVALは、順序名で修飾する必要があります。*sequence*.NEXTVALを参照すると、新しい順序番号が生成され、現行の順序番号がCURRVALに取得されます。

CURRVAL疑似列は、現行のユーザーが生成した順序番号を参照するために使用します。ただしCURRVALを参照する前に、NEXTVALを使用して、現行ユーザーのセッション内で順序番号を生成する必要があります。CURRVALは順序名で修飾します。*sequence*.CURRVALを参照すると、そのユーザーのプロセスに戻された最後の値が表示されます。

## NEXTVALおよびCURRVAL疑似列(続き)

### NEXTVALおよびCURRVALを使用するためのルール

NEXTVALおよびCURRVALは、次のようなコンテキストで使用できます。

- 副問合せには含まれてないSELECT文のSELECTリスト
- INSERT文内の副問合せのSELECTリスト
- INSERT文のVALUES句
- UPDATE文のSET句

NEXTVALおよびCURRVALは、次のようなコンテキストでは使用できません。

- ビューのSELECTリスト
- DISTINCTキーワードを含むSELECT文
- GROUP BY、HAVINGまたはORDER BY句を含むSELECT文
- SELECT、DELETEまたはUPDATE文内の副問合せ
- CREATE TABLEまたはALTER TABLE文内のDEFAULT式

詳細は、『Oracle Database SQL言語リファレンス11gリリース1(11.1)』の疑似列に関する節、およびCREATE SEQUENCEに関する節を参照してください。

Oracle Internal & Oracle Academy  
Use Only

## 順序の使用方法

- “Support”という新しい部門を所在地ID 2500に挿入します。

```
INSERT INTO departments (department_id,  
                          department_name, location_id)  
VALUES (dept_deptid_seq.NEXTVAL,  
        'Support', 2500);
```

1 rows inserted

- DEPT\_DEPTID\_SEQ順序の現行の値を表示します。

```
SELECT dept_deptid_seq.CURRVAL  
FROM dual;
```

ORACLE

Copyright © 2007, Oracle. All rights reserved.

### 順序の使用方法

このスライドの例では、DEPARTMENTS表に新しい部門を挿入しています。DEPT\_DEPTID\_SEQ順序を使用して、次のように新しい部門番号を生成します。

2番目のスライドの例に示すように、順序の現行の値は、*sequence\_name.CURRVAL*を使用して表示できます。この問合せの出力を次に示します。

	A Z	CURRVAL
1		120

この新しい部門のスタッフとして従業員を雇用するとします。すべての新しい従業員に対して実行するINSERT文には、次のコードを含めることができます。

```
INSERT INTO employees (employee_id, department_id, ...)  
VALUES (employees_seq.NEXTVAL, dept_deptid_seq .CURRVAL, ...);
```

**注意:** この例では、EMPLOYEE\_SEQという順序がすでに作成され、新しい従業員番号が生成されていることを前提としています。

## 順序値のキャッシュ

- 順序値をメモリーにキャッシュすると、その値へのアクセスが速くなります。
- 次の場合に、順序値のギャップが発生します。
  - ロールバックが発生した場合
  - システムがクラッシュした場合
  - 順序が別の表で使用されている場合

ORACLE

Copyright © 2007, Oracle. All rights reserved.

### 順序値のキャッシュ

順序をメモリーにキャッシュすると、順序値にすばやくアクセスできるようになります。キャッシュには、順序を初めて参照したときに移入されます。次の順序値の要求は、それぞれキャッシュされている順序から取得されます。最後の順序値が使用されると、次にその順序に対する要求があったときに、順序の別のキャッシュがメモリーに保存されます。

### 順序内のギャップ

順序はギャップがない状態で順序番号を発行しますが、このアクションはコミットまたはロールバックからは独立して実行されます。したがって、順序を含む文をロールバックすると、番号は失われます。

順序にギャップが生じる可能性のあるイベントとしては、他にシステム・クラッシュあります。順序の値がメモリーにキャッシュされた後にシステムがクラッシュすると、それらの値が失われます。

順序は表に直接関連付けられていないため、同じ順序を複数の表で使用できます。ただし、そうすると、それぞれの表で順序番号にギャップが生じる場合があります。



## 順序の変更

増分値、最大値、最小値、循環オプションまたはキャッシュ・オプションを変更します。

```
ALTER SEQUENCE dept_deptid_seq  
    INCREMENT BY 20  
    MAXVALUE 999999  
    NOCACHE  
    NOCYCLE;
```

```
ALTER SEQUENCE dept_deptid_seq succeeded.
```

ORACLE

Copyright © 2007, Oracle. All rights reserved.

### 順序の変更

順序のMAXVALUE制限に達すると、それ以上順序から値が割り当てられなくなり、順序がMAXVALUEを超えたことを示すエラーが戻されます。その順序を引き続き使用するためには、ALTER SEQUENCE文を使用して順序を変更します。

#### 構文

```
ALTER SEQUENCE sequence  
    [INCREMENT BY n]  
    [{MAXVALUE n | NOMAXVALUE}]  
    [{MINVALUE n | NOMINVALUE}]  
    [{CYCLE | NOCYCLE}]  
    [{CACHE n | NOCACHE}];
```

この構文で、*sequence*は順序の名前です。

詳細は、『Oracle Database SQL言語リファレンス11gリリース1(11.1)』のALTER SEQUENCEに関する節を参照してください。

## 順序変更のガイドライン

- 変更する順序の所有者であるか、ALTER権限を持っている必要があります。
- 変更が適用されるのは、それ以後の順序番号だけです。
- 異なる番号で順序を再開するには、順序を削除してから再作成する必要があります。
- いくつかの検証が実行されます。
- 順序を削除するには、DROP文を使用します。

```
DROP SEQUENCE dept_deptid_seq;  
DROP SEQUENCE dept_deptid_seq succeeded.
```

ORACLE

Copyright © 2007, Oracle. All rights reserved.

### 順序変更のガイドライン

- 順序を変更するには、その順序の所有者であるか、ALTER権限を持っている必要があります。順序を削除するには、所有者であるか、DROP ANY SEQUENCE権限を持っている必要があります。
- ALTER SEQUENCE文が適用されるのは、それ以後の順序番号だけです。
- START WITHオプションは、ALTER SEQUENCEを使用して変更することはできません。異なる番号で順序を再開するには、順序を削除してから再作成する必要があります。
- いくつかの検証が実行されます。たとえば、現行の順序番号より小さいMAXVALUEを新しく指定することはできません。

```
ALTER SEQUENCE dept_deptid_seq  
INCREMENT BY 20  
MAXVALUE 90  
NOCACHE  
NOCYCLE;
```

- 次のようなエラー・メッセージが表示されます。

```
Error report:  
SQL Error: ORA-04009: MAXVALUE cannot be made to be less than the current value  
04009. 00000 - "MAXVALUE cannot be made to be less than the current value"  
*Cause:      the current value exceeds the given MAXVALUE  
*Action:     make sure that the new MAXVALUE is larger than the current value
```

## 章の講義項目

- ビューの概要:
  - ビューのデータの作成、変更および取得
  - ビューでのDML操作
  - ビューの削除
- 順序の概要:
  - 順序の作成、使用および変更
  - 順序値のキャッシュ
  - NEXTVALおよびCURRVAL疑似列
- 索引の概要
  - 索引の作成および削除
- シノニムの概要
  - シノニムの作成および削除

ORACLE

Copyright © 2007, Oracle. All rights reserved.

# 索引

オブジェクト	説明
表	行で構成される、記憶域の基本単位。
ビュー	1つ以上の表のデータのサブセットを論理的に表します。
順序	数値を生成します。
索引	一部の問合せのパフォーマンスが向上します。
シノニム	オブジェクトに別名を付けます。

ORACLE

Copyright © 2007, Oracle. All rights reserved.

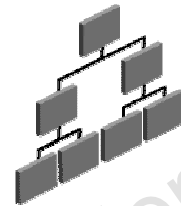
## 索引

索引は、一部の問合せのパフォーマンスを向上させるために作成できるデータベース・オブジェクトです。索引は、主キーまたは一意の制約を作成するときに、サーバーによって自動的に作成することもできます。

# 索引

## 索引の概要

- スキーマ・オブジェクトです。
- Oracleサーバーが、ポインタによって行の取得を高速化するために使用します。
- 高速なパス・アクセス手法によりデータをすばやく特定できるため、ディスクの入出力(I/O)が少なくなります。
- 索引の作成対象となる表とは独立しています。
- Oracleサーバーによって自動的に使用され、維持されます。



ORACLE

Copyright © 2007, Oracle. All rights reserved.

## 索引(続き)

Oracleサーバーの索引は、ポインタを使用することで行の取得を高速化できるスキーマ・オブジェクトです。索引は明示的にまたは自動的に作成できます。列に対して索引を設定しない場合は、全表スキャンが実行されます。

索引により、表内の行に対して直接かつ高速にアクセスできるようになります。その目的は、索引付けされたパスを使用してデータをすばやく特定し、ディスクI/Oを減らすことにあります。索引は、Oracleサーバーによって自動的に使用され、維持されます。索引が作成された後は、ユーザーによる直接的な操作が不要になります。

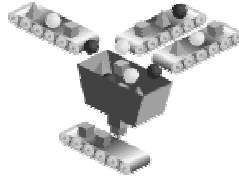
索引は、索引の作成対象となる表からは、論理的および物理的に独立しています。そのため、いつでも作成または削除でき、実表またはその他の索引にも影響しません。

**注意:** 表を削除すると、対応する索引も削除されます。

詳細は、『Oracle Database概要11gリリース1(11.1)』のスキーマ・オブジェクトの索引に関する節を参照してください。

## 索引の作成方法

- **自動:** 表の定義でPRIMARY KEYまたはUNIQUE制約を定義すると、一意の索引が自動的に作成されます。



- **手動:** ユーザーは列に対して一意でない索引を作成して、行に対するアクセスを高速にすることができます。



ORACLE

Copyright © 2007, Oracle. All rights reserved.

### 索引の作成方法

次の2つのタイプの索引を作成できます。

**一意の索引:** この索引は、表内の列にPRIMARY KEYまたはUNIQUE制約を定義すると、Oracle サーバーによって自動的に作成されます。索引の名前は、制約に付けた名前と同じになります。

**一意でない索引:** この索引は、ユーザーが作成できる索引です。たとえば、問合せでの結合用に FOREIGN KEY列に索引を作成すると、取得速度を向上させることができます。

**注意:** 一意の索引を手動で作成することもできますが、一意制約を作成して暗黙的に一意の索引が作成されるようにすることをお勧めします。

## 索引の作成

- 1つ以上の列に対して索引を作成します。

```
CREATE [UNIQUE][BITMAP]INDEX index  
ON table (column[, column]...);
```

- EMPLOYEES表のLAST\_NAME列に対する問合せアクセスの速度が向上します。

```
CREATE INDEX emp_last_name_idx  
ON employees (last_name);  
CREATE INDEX succeeded.
```

ORACLE

Copyright © 2007, Oracle. All rights reserved.

### 索引の作成

CREATE INDEX文を実行して、1つ以上の列に対して索引を作成します。

構文の内容

- index            索引の名前
- table            表の名前
- column          索引が作成される表内の列の名前

索引の作成対象となる1つまたは複数の列の値が一意でなければならないことを示すために、UNIQUEを指定します。各行に個別に索引を作成するのではなく、各個別キーのビットマップで索引が作成されなければならないことを示すには、BITMAPを指定します。ビットマップ索引によって、キー値にビットマップとして関連付けられたrowidsが格納されます。

詳細は、『Oracle Database SQL言語リファレンス11gリリース1(11.1)』のCREATE INDEXに関する節を参照してください。

# 索引作成のガイドライン

索引は次の場合に作成します。

✓	列に多様な値が含まれる場合
✓	列に多数のNULL値が含まれる場合
✓	1つ以上の列が頻繁にWHERE句または結合条件と合わせて使用される場合
✓	表の規模が大きく、ほとんどの問合せで表内の2～4%未満の行の取得が予想される場合

次の場合は索引を作成しないでください。

✗	列が問合せ内の条件としてあまり使用されない場合
✗	表の規模が小さいか、またはほとんどの問合せで表内の2～4%を超える行の取得が予想される場合
✗	表が頻繁に更新される場合
✗	索引付けされた列が式の一部として参照される場合

ORACLE

Copyright © 2007, Oracle. All rights reserved.

## 索引作成のガイドライン

### 索引の増加が必ずしも効果的ではない場合

表の索引の数を増やしても、問合せ速度が向上するわけではありません。索引付けされた表でDML操作を実行すると、各DML操作で索引の更新が必要になることを意味します。表に関連付けた索引の数が多くなると、DML操作後にすべての索引を更新するために、Oracleサーバーの負荷が高まることになります。

### 索引を作成する場合

このため、索引は次の場合にのみ作成してください。

- 列に多様な値が含まれる場合
- 列に多数のNULL値が含まれる場合
- 1つ以上の列が頻繁にWHERE句または結合条件と合わせて使用される場合
- 表の規模が大きく、ほとんどの問合せで2～4%未満の行の取得が予想される場合

一意性を確保するには、表の定義内に一意制約を定義します。それによって一意の索引が自動的に作成されます。



## 索引の削除

- データ・ディクショナリから索引を削除するには、DROP INDEX コマンドを使用します。

```
DROP INDEX index;
```

- データ・ディクショナリからemp\_last\_name\_idx索引を削除します。

```
DROP INDEX emp_last_name_idx;  
DROP INDEX emp_last_name_idx succeeded.
```

- 索引を削除するには、索引の所有者であるか、またはDROP ANY INDEX権限を持っている必要があります。

ORACLE

Copyright © 2007, Oracle. All rights reserved.

### 索引の削除

索引を変更することはできません。索引を変更するには、削除してから再作成する必要があります。

DROP INDEX文を実行して、データ・ディクショナリから索引の定義を削除します。索引を削除するには、索引の所有者であるか、またはDROP ANY INDEX権限を持っている必要があります。この構文で、*index*は索引の名前です。

**注意:** 表を削除すると、索引と制約が自動的に削除されますが、ビューと順序は残ります。

## 章の講義項目

- ビューの概要:
  - ビューのデータの作成、変更および取得
  - ビューでのDML操作
  - ビューの削除
- 順序の概要:
  - 順序の作成、使用および変更
  - 順序値のキャッシュ
  - NEXTVALおよびCURRVAL疑似列
- 索引の概要
  - 索引の作成および削除
- シノニムの概要
  - シノニムの作成および削除

ORACLE

Copyright © 2007, Oracle. All rights reserved.

## シノニム

オブジェクト	説明
表	行で構成される、記憶域の基本単位。
ビュー	1つ以上の表のデータのサブセットを論理的に表します。
順序	数値を生成します。
索引	一部の問合せのパフォーマンスが向上します。
シノニム	オブジェクトに別名を付けます。

ORACLE

Copyright © 2007, Oracle. All rights reserved.

### シノニム

シノニムは、別の名前で表を呼び出すことを可能にするデータベース・オブジェクトです。シノニムを作成することで、表に別名を付けることができます。

# オブジェクトのシノニムの作成

シノニム(オブジェクトの別名)を作成することで、オブジェクトに対するアクセスが容易になります。シノニムによって、次のことが可能になります。

- 別のユーザーが所有する表に対する参照をより容易に作成できます。
- オブジェクト名を短縮できます。

```
CREATE [PUBLIC] SYNONYM synonym
FOR      object;
```

ORACLE

Copyright © 2007, Oracle. All rights reserved.

## オブジェクトのシノニムの作成

別のユーザーが所有する表を参照するには、表の名前の前に、作成したユーザーの名前に続いてピリオドを付ける必要があります。シノニムを作成することで、オブジェクト名をスキーマで修飾する必要がなくなり、表、ビュー、順序、プロシージャまたはその他のオブジェクトの別名が得られます。この方法は、ビューなど、オブジェクト名が長い場合に特に便利です。

### 構文の内容

<b>PUBLIC</b>	すべてのユーザーがアクセスできるシノニムを作成します。
<i>synonym</i>	作成されるシノニムの名前です。
<i>object</i>	シノニムが作成されるオブジェクトを特定します。

### ガイドライン

- このオブジェクトをパッケージに含めることはできません。
- プライベート・シノニム名は、同じユーザーが所有するその他すべてのオブジェクトから区別されなければなりません。

詳細は、『Oracle Database SQL言語リファレンス11gリリース1(11.1)』のCREATE SYNONYMに関する節を参照してください。

## シノニムの作成および削除

- DEPT\_SUM\_VUビューの短縮名を作成します。

```
CREATE SYNONYM d_sum
FOR dept_sum_vu;
CREATE SYNONYM succeeded.
```

- シノニムを削除します。

```
DROP SYNONYM d_sum;
DROP SYNONYM d_sum succeeded.
```

ORACLE

Copyright © 2007, Oracle. All rights reserved.

### シノニムの作成および削除

#### シノニムの作成

このスライドの例では、素早く参照するためにDEPT\_SUM\_VUビューのシノニムを作成します。データベース管理者は、すべてのユーザーがアクセスできるパブリック・シノニムを作成できます。次の例では、Aliceが所有するDEPARTMENTS表に対して、DEPTというパブリック・シノニムを作成します。

```
CREATE SYNONYM succeeded.
```

```
CREATE PUBLIC SYNONYM dept
FOR alice.departments;
```

#### シノニムの削除

シノニムを削除するには、DROP SYNONYM文を使用します。パブリック・シノニムを削除できるのはデータベース管理者だけです。

```
DROP PUBLIC SYNONYM dept;
```

詳細は、『Oracle Database SQL言語リファレンス11gリリース1(11.1)』のDROP SYNONYMに関する節を参照してください。

## まとめ

この章では、次のことを学習しました。

- ビューの作成、使用および削除
- 順序を使用して順序番号を自動的に生成する方法
- 問合せの取得速度を向上させる索引の作成
- シノニムを使用してオブジェクトに別名を付ける方法

ORACLE

Copyright © 2007, Oracle. All rights reserved.

### まとめ

この章では、ビュー、順序、索引、シノニムなどのデータベース・オブジェクトについて学習しました。

## 演習11: パート2の概要

この演習では次の項目について説明しています。

- 順序の作成
- 順序の使用方法
- 一意でない索引の作成
- シノニムの作成

ORACLE

Copyright © 2007, Oracle. All rights reserved.

### 演習11: パート2の概要




この章の演習のパート2では、順序、索引およびシノニムの作成と使用に関する様々な演習に取り組みます。

この章の最後に示す、7から10の問題に答えます。

## 演習11

### パート1

1. HR部門のスタッフは、EMPLOYEES表の一部のデータを非表示にすることを必要としています。EMPLOYEES表の従業員番号、従業員名および部門番号に基づくEMPLOYEES\_VUというビューが必要です。従業員名のヘッダーをEMPLOYEEにしてください。
2. ビューが機能することを確認します。EMPLOYEES\_VUビューの内容を表示します。

	 EMPLOYEE_ID	 EMPLOYEE	 DEPARTMENT_ID
1	100	King	90
2	101	Kochhar	90
3	102	De Haan	90
4	103	Hunold	60
5	104	Ernst	60

...

19	205	Higgins	110
20	206	Gietz	110

3. このEMPLOYEES\_VUビューを使用して、すべての従業員名と部門番号を表示する、HR部門に対する問合せを記述します。

	 EMPLOYEE	 DEPARTMENT_ID
1	King	90
2	Kochhar	90
3	De Haan	90
4	Hunold	60
5	Ernst	60

...



19	Higgins	110
20	Gietz	110



## 演習11(続き)

4. 部門50は、従業員データにアクセスする必要があります。部門50のすべての従業員の従業員番号、従業員の姓、部門番号が表示される、DEPT50というビューを作成します。ビューの列にはEMPNO、EMPLOYEEおよびDEPTNOというラベルを付けるように依頼されています。セキュリティ上の理由から、ビューを通じて別の部門に従業員を再割当てする操作を許可しないでください。
5. DEPT50ビューの構造とコンテンツを表示します。

Name	Null	Type
-----	-----	-----
EMPNO	NOT NULL	NUMBER(6)
EMPLOYEE	NOT NULL	VARCHAR2(25)
DEPTNO		NUMBER(4)

	 EMPNO	 EMPLOYEE	 DEPTNO
1	124	Mourgos	50
2	141	Rajs	50
3	142	Davies	50
4	143	Matos	50
5	144	Vargas	50

6. ビューのテストを行います。部門80に対してMatosの再割当てを試してみます。

## 演習11(続き)

### パート2

7. DEPT表のPRIMARY KEY列で使える順序が必要です。この順序は200から開始し、最大値を1,000とします。順序が10ずつ増分されるようにします。順序にDEPT\_ID\_SEQという名前を付けます。
8. 順序のテストを行うために、DEPT表に2つの行を挿入するスクリプトを記述します。このスクリプトにlab\_11\_08.sqlという名前を付けます。このとき、ID列に作成した順序を使用してください。2つの部門 (EducationおよびAdministration)を追加します。追加した結果を確認します。スクリプト内のコマンドを実行します。
9. DEPT表のNAME列に一意ではない索引を作成します。
10. EMPLOYEES表のシノニムを作成し、EMPという名前を付けます。

Oracle Internal & Oracle Academy  
Use Only

---

A

## 演習の解答

---

Oracle Internal & Oracle Academy  
Use Only

## 演習 I の解答: はじめに

「はじめに」の演習の解答を次に示します。

### Oracle SQL Developer のデモンストレーション全体の実行: データベース接続の作成

1. 次のサイトにあるデモンストレーション「データベース接続の作成」にアクセスします。

[http://st-curriculum.oracle.com/tutorial/SQLDeveloper/html/module2/mod02\\_cp\\_newdbconn.htm](http://st-curriculum.oracle.com/tutorial/SQLDeveloper/html/module2/mod02_cp_newdbconn.htm)

### Oracle SQL Developer の起動

2. 「sqldeveloper」デスクトップ・アイコンを使用して、Oracle SQL Developer を起動します。

1. 「sqldeveloper」デスクトップ・アイコンをダブルクリックします。

**注意:** 初めて SQL Developer を起動するときは、java.exe ファイルのパスを指定する必要があります。これはすでに研修クラスのセットアップの一部として行われています。どのような場合も、入力するように指示されたら次のパスを入力してください。

D:\¥app¥Administrator¥product¥11.1.0¥client\_1¥jdevstudio¥jdk¥bin

### 新しい Oracle SQL Developer データベース接続の作成

3. 新しいデータベース接続を作成するには、接続ナビゲータで「Connections」を右クリックします。メニューから「New Connection」を選択します。「New/Select Database Connection」ダイアログ・ボックスが表示されます。
4. 次の情報を使用して、データベース接続を作成します。
  - a. 「Connection Name」: myconnection
  - b. 「Username」: oraxx。xx は、使用する PC の番号です。ora1～ora20 の範囲のアカウントから、ora アカウントを 1 つ割り当てるよう講師に依頼してください。
  - c. 「Password」: oraxx
  - d. 「Hostname」: データベース・サーバーが稼働しているマシンのホスト名を入力します。
  - e. 「Port」: 1521
  - f. 「SID」: ORCL
  - g. 「Save Password」チェック・ボックスを選択します。

## 演習 I の解答: はじめに(続き)

### Oracle SQL Developer データベース接続を使用したテストと接続

5. 新しい接続をテストします。
  1. 「New/Select Database Connection」ウィンドウの「Test」ボタンをクリックします。
6. 状態が「Success」の場合は、この新しい接続を使用してデータベースに接続します。
  1. 状態が「Success」の場合は、「Connect」ボタンをクリックします。

### 接続ナビゲータでの表の参照

7. 接続ナビゲータの「Tables」ノードで、使用可能なオブジェクトを表示します。次の表が存在することを確認します。

COUNTRIES  
DEPARTMENTS  
EMPLOYEES  
JOB\_GRADES  
JOB\_HISTORY  
JOBS

LOCATIONS  
REGIONS

1. 横にあるプラス記号をクリックして、myconnection 接続を展開します。
  2. 横にあるプラス記号をクリックして、「Tables」アイコンを展開します。
8. EMPLOYEES 表の構造を参照します。
    1. EMPLOYEES 表の構造を表示するには、EMPLOYEES をクリックします。デフォルトでは「Columns」タブがアクティブになり、表の構造が表示されます。
  9. DEPARTMENTS 表のデータを表示します。
    1. DEPARTMENTS 表をクリックします。
    2. 「Data」タブをクリックします。表のデータが表示されます。

## 演習 I の解答: はじめに(続き)

### SQL Worksheet の表示

10. 新しい SQL Worksheet をオープンします。SQL Worksheet 用に使用できるショートカット・アイコンを確認します。

1. 新しい SQL Worksheet をオープンするには、「Tools」メニューから「SQL Worksheet」を選択します。
2. または、myconnection を右クリックし、「SQL Worksheet」を選択します。
3. SQL Worksheet のショートカット・アイコンを確認します。具体的には、「Execute Statement」アイコンと「Run Script」アイコンを検索します。

Oracle Internal & Oracle Academy  
Use Only

## 演習 1 の解答: SQL SELECT 文を使用したデータの取得

### パート 1

知識について確認します。

1. 次の SELECT 文は正しく実行されます。

```
SELECT last_name, job_id, salary AS Sal
FROM   employees;
```

○/×

2. 次の SELECT 文は正しく実行されます。

```
SELECT *
FROM   job_grades;
```

○/×

3. 次の文には 4 つのコーディング・エラーがあります。特定できますか?

```
SELECT      employee_id, last_name
sal x 12    ANNUAL SALARY
FROM        employees;
```

- EMPLOYEES 表には sal という列はありません。列の名前は SALARY になっています。
- 行 2 の乗算演算子は、x ではなく\*です。
- 別名 ANNUAL SALARY にスペースを含めることはできません。この別名は ANNUAL\_SALARY にするか、二重引用符で囲む必要があります。
- LAST\_NAME 列の後にカンマを入れる必要があります。

### パート 2

演習を始める前に次の点を確認します。

- すべての演習ファイルを D:\labs\SQL1\labs に保存します。
- SQL 文を SQL Worksheet に入力します。SQL Developer にスクリプトを保存するには、「File」メニューから「Save As」を選択するか、SQL Worksheet を右クリックして「Save file」を選択し、SQL 文を lab\_<lessonno>\_<stepno>.sql スクリプトとして保存します。既存のスクリプトを変更するには、「File」→「Open」を選択してスクリプト・ファイルを開き、変更を加えた後、「Save As」を選択して異なるファイル名で保存します。
- 問合せを実行するには、SQL Worksheet で「Execute Statement」アイコンをクリックするか、[F9]を押します。DML 文および DDL 文の場合は、「Run Script」アイコンをクリックするか、[F5]を押します。
- 保存されているスクリプトを実行した後は、同じワークシートに次の問合せを入力しないようにします。新しいワークシートを開いてください。

## 演習 1 の解答: SQL SELECT 文を使用したデータの取得(続き)

あなたは Acme Corporation の SQL プログラマとして採用されました。最初の仕事は、人事管理表のデータに基づいたレポートの作成です。

4. まず、DEPARTMENTS 表の構造とその内容を確認します。

a. DEPARTMENTS 表の構造を特定するには、次のコマンドを実行します。

```
DESCRIBE departments
```

b. DEPARTMENTS 表に含まれるデータを表示するには、次のコマンドを実行します。

```
SELECT *  
FROM departments;
```

5. EMPLOYEES 表の構造を確認する必要があります。

```
DESCRIBE employees
```

HR 部門は、各従業員の姓、職務コード、雇用日、従業員番号を表示する問合せを必要としています。また、従業員番号を最初に表示する必要があります。HIRE\_DATE 列に別名 STARTDATE を指定します。SQL 文を、lab\_01\_05.sql という名前のファイルに保存して、このファイルを HR 部門にディスパッチできるようにします。

```
SELECT employee_id, last_name, job_id, hire_date StartDate  
FROM employees;
```

6. lab\_01\_05.sql ファイルで問合せをテストして、正しく実行されることを確認します。

```
SELECT employee_id, last_name, job_id, hire_date StartDate  
FROM employees;
```

7. HR 部門は、EMPLOYEES 表から一意の職務コードをすべて表示する問合せを必要としています。

```
SELECT DISTINCT job_id  
FROM employees;
```



## 演習 1 の解答: SQL SELECT 文を使用したデータの取得(続き)

### パート 3

時間があるときは、次の演習問題に進みます。

8. HR 部門は、従業員に関するそのレポートに、さらにわかりやすい列ヘッダーを使用するように求めています。文を lab\_01\_05.sql から新しい SQL Worksheet にコピーします。列ヘッダーにそれぞれ Emp #、Employee、Job、Hire Date という名前を付けます。その後、問合せを再度実行します。

```
SELECT employee_id "Emp #", last_name "Employee",  
       job_id "Job", hire_date "Hire Date"  
FROM   employees;
```

9. HR 部門は、すべての従業員とその職務IDのレポートを要求しています。姓に職務IDを(カンマおよび空白で区切って)連結して表示し、列に Employee and Title という名前を付けます。

```
SELECT last_name||', '||job_id "Employee and Title"  
FROM   employees;
```

さらに演習を続ける場合は、次の演習問題に進みます。

10. EMPLOYEES 表のデータを理解するために、EMPLOYEES 表のすべてのデータを表示する問合せを作成します。それぞれの列の出力をカンマで区切ります。列タイトルに THE\_OUTPUT という名前を付けます。

```
SELECT employee_id || ',' || first_name || ',' || last_name  
       || ',' || email || ',' || phone_number || ',' || job_id  
       || ',' || manager_id || ',' || hire_date || ',' ||  
       || salary || ',' || commission_pct || ',' || department_id  
       THE_OUTPUT  
FROM   employees;
```

## 演習 2 の解答: データの制限とソート

HR 部門から、いくつかの問合せの作成支援を依頼されました。

1. HR 部門では、予算上の問題で、給与が\$12,000 を超える従業員の姓と給与を表示するレポートを必要としています。作成した SQL 文は、lab\_02\_01.sql という名前のファイルとして保存します。この問合せを実行してください。

```
SELECT last_name, salary
FROM employees
WHERE salary > 12000;
```

2. 新しい SQL Worksheet を開きます。従業員番号 176 の姓と部門番号を表示するレポートを作成します。

```
SELECT last_name, department_id
FROM employees
WHERE employee_id = 176;
```

3. HR 部門は、給与の高い従業員と低い従業員を検索する必要があります。lab\_02\_01.sql を変更して、給与が\$5,000～\$12,000 の範囲にない従業員の姓と給与を表示するようにします。作成した SQL 文を、lab\_02\_03.sql として保存してください。

```
SELECT last_name, salary
FROM employees
WHERE salary NOT BETWEEN 5000 AND 12000;
```

4. 姓が Matos および Taylor である従業員の姓、職務 ID および開始日を表示するレポートを作成します。開始日に基づいて、問合せを昇順でソートしてください。

```
SELECT last_name, job_id, hire_date
FROM employees
WHERE last_name IN ('Matos', 'Taylor')
ORDER BY hire_date;
```

5. 部門 20 または 50 に所属するすべての従業員の姓と部門番号を、名前のアルファベット順 (昇順) で表示します。

```
SELECT last_name, department_id
FROM employees
WHERE department_id IN (20, 50)
ORDER BY last_name ASC;
```

## 演習 2 の解答: データの制限とソート(続き)

6. lab\_02\_03.sql を変更して、給与が \$5,000 ~ \$12,000 の範囲にあり部門 20 または 50 に所属している従業員の姓と給与を表示するようにします。列にそれぞれ Employee および Monthly Salary というラベルを付けます。lab\_02\_03.sql は、lab\_02\_06.sql として保存し直します。lab\_02\_06.sql の文を実行してください。

```
SELECT    last_name "Employee", salary "Monthly Salary"
FROM      employees
WHERE     salary BETWEEN 5000 AND 12000
AND       department_id IN (20, 50);
```

7. HR 部門は、1994 年に雇用されたすべての従業員の姓と雇用日を表示するレポートを必要としています。

```
SELECT    last_name, hire_date
FROM      employees
WHERE     hire_date LIKE '%94';
```

8. 担当マネージャがいないすべての従業員の姓と職務を表示するレポートを作成します。

```
SELECT    last_name, job_id
FROM      employees
WHERE     manager_id IS NULL;
```

9. 歩合を受け取るすべての従業員の姓、給与および歩合を表示するレポートを作成します。データを、給与および歩合の降順でソートします。ORDER BY 句で、列の位置を数値で指定してください。

```
SELECT    last_name, salary, commission_pct
FROM      employees
WHERE     commission_pct IS NOT NULL
ORDER BY 2 DESC, 3 DESC;
```

10. HR 部門のメンバーは、作成中の問合せにさらに柔軟性を求めています。求められる柔軟性とは、レポートで、プロンプト表示後にユーザーが指定する額を超える給与の従業員の姓と給与が表示されるようにすることです。この問合せを lab\_02\_10.sql という名前のファイルに保存してください。プロンプトが表示されたときに 12000 を入力すると、レポートに次の結果が表示されます。(演習 1 の実習で作成した問合せを変更して使用できます。)この問合せを lab\_02\_10.sql という名前のファイルに保存します。

- a. ダイアログ・ボックスに値を入力するプロンプトが表示されたら、12000 と入力します。「OK」をクリックします。

```
SELECT    last_name, salary
FROM      employees
WHERE     salary > &sal_amt;
```

## 演習 2 の解答: データの制限とソート(続き)

11. HR 部門では、マネージャに基づいたレポートを実行する必要があります。プロンプトを表示してユーザーにマネージャ ID の入力を求め、そのマネージャが管理する従業員の従業員 ID、姓、給与および部門を生成する問合せを作成します。HR 部門は、選択した列でレポートをソートする機能を必要としています。データのテストには、次の値を使用できます。

manager\_id = 103、last\_name によってソート

manager\_id = 201、salary によってソート

manager\_id = 124、employee\_id によってソート

```
SELECT employee_id, last_name, salary, department_id
FROM employees
WHERE manager_id = &mgr_num
ORDER BY &order_col;
```

時間があるときは、次の演習問題に進みます。

12. 姓の 3 文字目が“a”であるすべての従業員の姓を表示します。

```
SELECT last_name
FROM employees
WHERE last_name LIKE '__a%';
```

13. 姓の中に“a”と“e”の両方が含まれるすべての従業員の姓を表示します。

```
SELECT last_name
FROM employees
WHERE last_name LIKE '%a%'
AND last_name LIKE '%e%';
```

さらに演習を続ける場合は、次の演習問題に進みます。

14. 職務が販売担当者または在庫管理者であり、給与が\$2,500、\$3,500 または\$7,000 と等しくないすべての従業員の姓、職務および給与を表示します。

```
SELECT last_name, job_id, salary
FROM employees
WHERE job_id IN ('SA_REP', 'ST_CLERK')
AND salary NOT IN (2500, 3500, 7000);
```

15. lab\_02\_06.sql を変更して、歩合が 20%であるすべての従業員の姓、給与および歩合を表示するようにします。lab\_02\_06.sql を lab\_02\_15.sql として保存し直します。lab\_02\_15.sql の文を再実行してください。

```
SELECT last_name "Employee", salary "Monthly Salary",
commission_pct
FROM employees
WHERE commission_pct = .20;
```

### 演習 3 の解答: 単一行関数を使用した出力のカスタマイズ

1. システムの日付を表示する問合せを記述します。列に Date というラベルを付けます。

**注意:** 使用するデータベースがリモートの異なるタイムゾーンにある場合、出力は、データベースが存在するオペレーティング・システムの日付になります。

```
SELECT    sysdate "Date"
FROM      dual;
```

2. HR 部門は、各従業員の従業員番号、姓、給与、15.5%増額された給与(整数で表示)を表示するレポートを必要としています。列に New Salary というラベルを付けます。作成した SQL 文を lab\_03\_02.sql という名前のファイルに保存します。

```
SELECT    employee_id, last_name, salary,
          ROUND(salary * 1.155, 0) "New Salary"
FROM      employees;
```

3. lab\_03\_02.sql ファイルで問合せを実行します。

```
SELECT    employee_id, last_name, salary,
          ROUND(salary * 1.155, 0) "New Salary"
FROM      employees;
```

4. 問合せ lab\_03\_02.sql を変更して、新しい給与から前の給与を引いた列を追加します。列に Increase というラベルを付けます。ファイルの内容を lab\_03\_04.sql として保存します。修正した問合せを実行します。

```
SELECT    employee_id, last_name, salary,
          ROUND(salary * 1.155, 0) "New Salary",
          ROUND(salary * 1.155, 0) - salary "Increase"
FROM      employees;
```

5. 名前が文字「J」、「A」または「M」で始まるすべての従業員の姓(最初の文字は大文字、その他のすべての文字は小文字)と姓の長さを表示する問合せを記述します。それぞれの列に適切なラベルを付けます。結果を、従業員の姓でソートします。

```
SELECT    INITCAP(last_name) "Name",
          LENGTH(last_name) "Length"
FROM      employees
WHERE     last_name LIKE 'J%'
OR        last_name LIKE 'M%'
OR        last_name LIKE 'A%'
ORDER BY last_name ;
```

### 演習 3 の解答: 単一行関数を使用した出力のカスタマイズ(続き)

問合せを記述し直して、姓の先頭に付く文字の入力を求めるプロンプトが表示されるようにします。たとえば、文字の入力を求められたときにユーザーが「H」(大文字)を入力した場合に、姓が「H」で始まるすべての従業員が出力に表示されるようにします。

```
SELECT  INITCAP(last_name) "Name",
        LENGTH(last_name) "Length"
FROM    employees
WHERE   last_name LIKE '&start_letter%'
ORDER BY last_name;
```

問合せを変更して、入力した文字の大文字と小文字の区別が出力に影響しないようにします。入力した文字は、SELECT 問合せによって処理される前に、大文字に変換されている必要があります。

```
SELECT  INITCAP(last_name) "Name",
        LENGTH(last_name) "Length"
FROM    employees
WHERE   last_name LIKE UPPER('&start_letter%' )
ORDER BY last_name;
```

6. HR 部門は、それぞれの従業員の雇用期間を確認する必要があります。従業員ごとに姓を表示して、その従業員が雇用された日付から今日までの月数を計算します。列に MONTHS\_WORKED というラベルを付けます。結果を雇用月数の順に並べます。月数を最も近い整数に丸めます。

**注意:** この問合せは実行された日付に依存するため、MONTHS\_WORKED 列には異なる値が表示されます。

```
SELECT last_name, ROUND(MONTHS_BETWEEN(
        SYSDATE, hire_date)) MONTHS_WORKED
FROM    employees
ORDER BY months_worked;
```

時間があるときは、次の演習問題に進みます。

7. すべての従業員の姓と給与を表示する問合せを作成します。給与を 15 文字長に書式設定し、左側に \$ 記号を埋め込みます。列に SALARY というラベルを付けます。

```
SELECT last_name,
        LPAD(salary, 15, '$') SALARY
FROM    employees;
```

### 演習 3 の解答: 単一行関数を使用した出力のカスタマイズ(続き)

8. 従業員の姓の最初の 8 文字を表示し、その給与額をアスタリスクで示す問合せを作成します。各アスタリスクは、1,000 ドルを表します。データを給与の降順でソートします。列に EMPLOYEES\_AND\_THEIR\_SALARIES というラベルを付けます。

```
SELECT rpad(last_name, 8)||' '||  
       rpad(' ', salary/1000+1, '*')  
       EMPLOYEES_AND_THEIR_SALARIES  
FROM   employees  
ORDER BY salary DESC;
```

9. 部門 90 のすべての従業員の姓と雇用週数を表示する問合せを作成します。週数の列に TENURE というラベルを付けます。週数の小数点をゼロに切り捨てます。レコードを、従業員の在職期間の降順で表示します。
- a. **注意:** TENURE 値は、問合せを実行する日付によって異なります。

```
SELECT last_name, trunc((SYSDATE-hire_date)/7) AS TENURE  
FROM   employees  
WHERE  department_id = 90  
ORDER BY TENURE DESC
```

## 演習 4 の解答: 変換関数と条件式の使用方法

1. 各従業員の次のデータを生成するレポートを作成します。  
<employee last name> earns <salary> monthly but wants <3 times salary.>。列に Dream Salaries というラベルを付けます。

```
SELECT last_name || ' earns '  
      || TO_CHAR(salary, 'fm$99,999.00')  
      || ' monthly but wants '  
      || TO_CHAR(salary * 3, 'fm$99,999.00')  
      || '. ' "Dream Salaries"  
FROM   employees;
```

2. 各従業員の姓、雇用日、および 6 か月の雇用期間後の最初の月曜日となる給与審査日を表示します。列に REVIEW というラベルを付けます。「Monday, the Thirty-First of July, 2000」のような書式で表示されるように日付の書式を設定します。

```
SELECT last_name, hire_date,  
       TO_CHAR(NEXT_DAY(ADD_MONTHS(hire_date, 6), 'MONDAY'),  
              'fmDay, "the" Ddspth "of" Month, YYYY') REVIEW  
FROM   employees;
```

3. 従業員の姓、雇用日および従業員が勤務を開始した曜日を表示します。列に DAY というラベルを付けます。月曜日から開始されるように、曜日で結果をソートします。

```
SELECT last_name, hire_date,  
       TO_CHAR(hire_date, 'DAY') DAY  
FROM   employees  
ORDER BY TO_CHAR(hire_date - 1, 'd');
```

4. 従業員の姓と歩合の額を表示する問合せを作成します。従業員が歩合を受け取らない場合は、「No Commission」を表示します。列に COMM というラベルを付けます。

```
SELECT last_name,  
       NVL(TO_CHAR(commission_pct), 'No Commission') COMM  
FROM   employees;
```



#### 演習 4 の解答: 変換関数と条件式の使用法 (続き)

5. DECODE 関数で次のデータを使用して、列 JOB\_ID の値に基づくすべての従業員の等級を表示する問合せを作成します。

職務	等級
AD_PRES	A
ST_MAN	B
IT_PROG	C
SA_REP	D
ST_CLERK	E
None of the above	0

```
SELECT job_id, decode (job_id,
                        'ST_CLERK', 'E',
                        'SA_REP', 'D',
                        'IT_PROG', 'C',
                        'ST_MAN', 'B',
                        'AD_PRES', 'A',
                        '0') GRADE
FROM employees;
```

6. CASE 構文を使用して、前の演習問題の文を記述し直します。

```
SELECT job_id, CASE job_id
                WHEN 'ST_CLERK' THEN 'E'
                WHEN 'SA_REP' THEN 'D'
                WHEN 'IT_PROG' THEN 'C'
                WHEN 'ST_MAN' THEN 'B'
                WHEN 'AD_PRES' THEN 'A'
                ELSE '0' END GRADE
FROM employees;
```

## 演習 5 の解答: グループ関数を使用した集計データのレポート

次の 3 つの文が正しいかどうかを判断してください。○または×を円で囲みます。

1. グループ関数は複数の行に対して動作し、グループごとに 1 つの結果を返します。

○/×

2. グループ関数では計算に NULL が含まれます。

○/×

3. WHERE 句は、グループ計算に取り込む前に行を制限します。

○/×

HR 部門は次のレポートを必要としています。

4. すべての従業員の給与の最高額、最低額、合計および平均を求めます。列にそれぞれ、Maximum、Minimum、Sum、Average というラベルを付けます。結果を最も近い整数に丸めます。作成した SQL 文を lab\_05\_04.sql として保存します。問合せを実行します。

```
SELECT ROUND(MAX(salary),0) "Maximum",  
       ROUND(MIN(salary),0) "Minimum",  
       ROUND(SUM(salary),0) "Sum",  
       ROUND(AVG(salary),0) "Average"  
FROM   employees;
```

5. 職種ごとの給与の最低額、最高額、合計および平均を表示するように、lab\_05\_04.sql の問合せを変更します。lab\_05\_04.sql を lab\_05\_05.sql として保存し直します。lab\_05\_05.sql の文を実行します。

```
SELECT job_id, ROUND(MAX(salary),0) "Maximum",  
       ROUND(MIN(salary),0) "Minimum",  
       ROUND(SUM(salary),0) "Sum",  
       ROUND(AVG(salary),0) "Average"  
FROM   employees  
GROUP BY job_id;
```

6. 同じ職務の従業員の数を表示する問合せを記述します。

```
SELECT job_id, COUNT(*)  
FROM   employees  
GROUP BY job_id;
```

HR 部門のユーザーに職種の入力を求めるように、問合せを汎用化します。スクリプトを lab\_05\_06.sql という名前のファイルに保存します。問合せを実行します。入力を求められたら、IT\_PROG と入力します。「OK」をクリックします。

```
SELECT job_id, COUNT(*)  
FROM   employees  
WHERE  job_id = '&job_title'  
GROUP BY job_id;
```

## 演習 5 の解答: グループ関数を使用した集計データのレポート(続き)

7. マネージャをリストせずにその数を特定します。列に Number of Managers というラベルを付けます。

ヒント: MANAGER\_ID 列を使用して、マネージャの数を特定します。

```
SELECT COUNT(DISTINCT manager_id) "Number of Managers"
FROM   employees;
```

8. 給与の最高額と最低額の差額を求めます。列に DIFFERENCE というラベルを付けます。

```
SELECT   MAX(salary) - MIN(salary) DIFFERENCE
FROM     employees;
```

時間があるときは、次の演習問題に進みます。

9. マネージャのマネージャ番号と、そのマネージャが管理する最も給与の低い従業員の給与を表示するレポートを作成します。マネージャが不明な従業員は除外します。また、給与の最低額が\$6,000 以下のグループは除外します。出力を給与の降順でソートします。

```
SELECT   manager_id, MIN(salary)
FROM     employees
WHERE    manager_id IS NOT NULL
GROUP BY manager_id
HAVING   MIN(salary) > 6000
ORDER BY MIN(salary) DESC;
```

さらに演習を続ける場合は、次の演習問題に進みます。

10. 従業員の総数と、1995 年、1996 年、1997 年および 1998 年に雇用された従業員の数を表示する問合せを作成します。適切な列ヘッダーを作成します。

```
SELECT   COUNT(*) total,
          SUM(DECODE(TO_CHAR(hire_date, 'YYYY'), 1995, 1, 0)) "1995",
          SUM(DECODE(TO_CHAR(hire_date, 'YYYY'), 1996, 1, 0)) "1996",
          SUM(DECODE(TO_CHAR(hire_date, 'YYYY'), 1997, 1, 0)) "1997",
          SUM(DECODE(TO_CHAR(hire_date, 'YYYY'), 1998, 1, 0)) "1998"
FROM     employees;
```

## 演習 5 の解答: グループ関数を使用した集計データのレポート(続き)

11. 部門 20、50、80 および 100 に対して、職務、部門番号に基づくその職務の給与、およびその職務の給与の合計を表示するマトリクス形式の問合せを作成して、それぞれの列に適切なヘッダーを付けます。

```
SELECT    job_id "Job",
          SUM(DECODE(department_id , 20, salary)) "Dept 20",
          SUM(DECODE(department_id , 50, salary)) "Dept 50",
          SUM(DECODE(department_id , 80, salary)) "Dept 80",
          SUM(DECODE(department_id , 90, salary)) "Dept 90",
          SUM(salary) "Total"
FROM      employees
GROUP BY  job_id;
```

Oracle Internal & Oracle Academy  
Use Only

## 演習 6 の解答: 複数の表のデータの表示

1. HR 部門を検索して、すべての部門の住所を出力する問合せを記述します。LOCATIONS 表と COUNTRIES 表を使用します。出力には、所在地 ID、番地、市、州または県、および国を表示します。結果の出力には、NATURAL JOIN を使用します。

```
SELECT location_id, street_address, city, state_province, country_name
FROM   locations
NATURAL JOIN countries;
```

2. HR 部門はすべての従業員のレポートを必要としています。すべての従業員の姓、部門番号および部門名を表示する問合せを記述します。

```
SELECT last_name, department_id, department_name
FROM   employees
JOIN   departments
USING (department_id);
```

3. HR 部門はトロントの従業員のレポートを必要としています。トロントで勤務しているすべての従業員の姓、職務、部門番号および部門名を表示します。

```
SELECT e.last_name, e.job_id, e.department_id, d.department_name
FROM   employees e JOIN departments d
ON     (e.department_id = d.department_id)
JOIN   locations l
ON     (d.location_id = l.location_id)
WHERE  LOWER(l.city) = 'toronto';
```

4. 従業員の姓と従業員番号および担当マネージャの姓とマネージャ番号を表示するレポートを作成します。それぞれの列には Employee、Emp#、Manager および Mgr# のラベルを付けます。作成した SQL 文を lab\_06\_04.sql として保存します。問合せを実行します。

```
SELECT w.last_name "Employee", w.employee_id "EMP#",
       m.last_name "Manager", m.employee_id  "Mgr#"
FROM   employees w join employees m
ON     (w.manager_id = m.employee_id);
```

5. 担当マネージャのいない King を含め、すべての従業員が表示されるように lab\_06\_04.sql を変更します。結果は、従業員番号の順に並べます。作成した SQL 文を lab\_06\_05.sql として保存します。lab\_06\_05.sql の問合せを実行します。

```
SELECT w.last_name "Employee", w.employee_id "EMP#",
       m.last_name "Manager", m.employee_id  "Mgr#"
FROM   employees w
LEFT   OUTER JOIN employees m
ON     (w.manager_id = m.employee_id)
ORDER BY 2;
```

## 演習 6 の解答: 複数の表のデータの表示 (続き)

6. HR 部門用のレポートを作成して、従業員の姓、部門番号、および指定した従業員と同じ部門に勤務するすべての従業員を表示します。それぞれの列に、該当するラベルを付けます。作成したスクリプトを lab\_06\_06.sql という名前でファイルに保存します。問合せを実行します。

```
SELECT e.department_id department, e.last_name employee,  
       c.last_name colleague  
FROM   employees e JOIN employees c  
ON      (e.department_id = c.department_id)  
WHERE  e.employee_id <> c.employee_id  
ORDER BY e.department_id, e.last_name, c.last_name;
```

7. HR 部門では、職務等級と給与に関するレポートを必要としています。JOB\_GRADES 表を理解するために、まず、JOB\_GRADES 表の構造を確認します。次に、すべての従業員の名前、職務、部門名、給与および等級を表示する問合せを作成します。

```
DESC JOB_GRADES  
  
SELECT e.last_name, e.job_id, d.department_name,  
       e.salary, j.grade_level  
FROM   employees e JOIN departments d  
ON      (e.department_id = d.department_id)  
JOIN    job_grades j  
ON      (e.salary BETWEEN j.lowest_sal AND j.highest_sal);
```

さらに演習を続ける場合は、次の演習問題に進みます。

8. HR 部門では、Davies より後に雇用されたすべての従業員の名前を調べています。従業員 Davies より後に雇用されたすべての従業員の名前と雇用日を表示する問合せを作成します。

```
SELECT e.last_name, e.hire_date  
FROM   employees e JOIN employees davies  
ON      (davies.last_name = 'Davies')  
WHERE  davies.hire_date < e.hire_date;
```

9. HR 部門では、担当マネージャより前に雇用されたすべての従業員の名前と雇用日、およびその担当マネージャの名前と雇用日を調べる必要があります。作成したスクリプトを lab\_06\_09.sql という名前でファイルに保存します。

```
SELECT w.last_name, w.hire_date, m.last_name, m.hire_date  
FROM   employees w JOIN employees m  
ON      (w.manager_id = m.employee_id)  
WHERE  w.hire_date < m.hire_date;
```

## 演習 7 の解答: 副問合せによる問合せの解決方法

1. HR 部門は、ユーザーに従業員の姓の入力を求めるプロンプトを表示し、入力された名前の従業員と同じ部門に所属するすべての従業員(入力した従業員は除く)の姓と雇用日を表示する問合せを必要としています。たとえば、ユーザーが Zlotkey と入力すると、Zlotkey とともに勤務するすべての従業員(Zlotkey は除く)が検索されます。

```
UNDEFINE Enter_name

SELECT last_name, hire_date
FROM   employees
WHERE  department_id = (SELECT department_id
                        FROM   employees
                        WHERE  last_name = '&&Enter_name')
AND    last_name <> '&Enter_name';
```

2. 給与が平均給与より多いすべての従業員の従業員番号、姓および給与を表示するレポートを作成してください。給与の昇順に結果をソートしてください。

```
SELECT employee_id, last_name, salary
FROM   employees
WHERE  salary > (SELECT AVG(salary)
                 FROM   employees)
ORDER BY salary;
```

3. 姓に“u”の文字が含まれる従業員が所属する部門に勤務するすべての従業員の従業員番号と姓を表示する問合せを記述します。作成した SQL 文は、lab\_07\_03.sql として保存します。この問合せを実行してください。

```
SELECT employee_id, last_name
FROM   employees
WHERE  department_id IN (SELECT department_id
                        FROM   employees
                        WHERE  last_name like '%u%');
```

4. HR 部門では、部門の所在地 ID が 1700 であるすべての従業員の姓、部門番号および職務 ID を表示するレポートを必要としています。

```
SELECT last_name, department_id, job_id
FROM   employees
WHERE  department_id IN (SELECT department_id
                        FROM   departments
                        WHERE  location_id = 1700);
```

## 演習 7 の解答: 副問合せによる問合せの解決方法(続き)

問合せを変更して、ユーザーに所在地 ID の入力を求めるプロンプトが表示されるようにします。この問合せは、lab\_07\_04.sql という名前のファイルに保存します。

```
SELECT last_name, department_id, job_id
FROM   employees
WHERE  department_id IN (SELECT department_id
                        FROM   departments
                        WHERE  location_id = &Enter_location);
```

5. HR 用に、King を上司とするすべての従業員の姓と給与を表示するレポートを作成します。

```
SELECT last_name, salary
FROM   employees
WHERE  manager_id = (SELECT employee_id
                    FROM   employees
                    WHERE  last_name = 'King');
```

6. HR 用に、Executive 部門に所属するすべての従業員の部門番号、姓および職務 ID を表示するレポートを作成します。

```
SELECT department_id, last_name, job_id
FROM   employees
WHERE  department_id IN (SELECT department_id
                        FROM   departments
                        WHERE  department_name = 'Executive');
```

時間があるときは、次の演習問題に進みます。

7. lab\_07\_03.sql の問合せを変更し、給与が平均給与より多い従業員で、姓に“u”の文字が含まれる従業員が所属する部門に勤務するすべての従業員の従業員番号、姓および給与が表示されるようにしてください。lab\_07\_03.sql を lab\_07\_07.sql として保存し直します。lab\_07\_07.sql の文を実行してください。

```
SELECT employee_id, last_name, salary
FROM   employees
WHERE  department_id IN (SELECT department_id
                        FROM   employees
                        WHERE  last_name like '%u%')
AND    salary > (SELECT AVG(salary)
                FROM   employees);
```



## 演習 8 の解答: 集合演算子の使用方法

1. HR 部門では、職務 ID の ST\_CLERK が含まれない部門の部門 ID のリストを必要としています。集合演算子を使用してこのレポートを作成してください。

```
SELECT department_id
FROM departments
MINUS
SELECT department_id
FROM employees
WHERE job_id = 'ST_CLERK';
```

2. HR 部門では、部門が存在しない国のリストを必要としています。それらの国の国 ID と国名を表示します。集合演算子を使用してこのレポートを作成してください。

```
SELECT country_id, country_name
FROM countries
MINUS
SELECT l.country_id, c.country_name
FROM locations l JOIN countries c
ON (l.country_id = c.country_id)
JOIN departments d
ON d.location_id=l.location_id;
```

3. 部門 10、50 および 20 に対する職務のリストをこの順に出力します。集合演算子を使用して職務 ID と部門 ID を表示してください。

```
SELECT distinct job_id, department_id
FROM employees
WHERE department_id = 10
UNION ALL
SELECT DISTINCT job_id, department_id
FROM employees
WHERE department_id = 50
UNION ALL
SELECT DISTINCT job_id, department_id
FROM employees
WHERE department_id = 20
```

4. 現行の職務が会社に雇用された当初の職務と同じ(つまり、職務を異動したが、現在は元の職務に戻っている)従業員の従業員 ID と職務 ID のリストを表示するレポートを作成します。

```
SELECT employee_id, job_id
FROM employees
INTERSECT
SELECT employee_id, job_id
FROM job_history;
```

## 演習 8 の解答: 集合演算子の使用方法(続き)

5. HR 部門では、次の仕様のレポートを必要としています。

- EMPLOYEES 表のすべての従業員の姓と部門 ID (部門に所属しているかどうかに関係なく)
- DEPARTMENTS 表のすべての部門の部門 ID と部門名 (所属する従業員が存在するかどうかに関係なく)

複合問合せを記述して、この仕様を実現してください。

```
SELECT last_name, department_id, TO_CHAR(null)
FROM   employees
UNION
SELECT TO_CHAR(null), department_id, department_name
FROM   departments;
```

Oracle Internal & Oracle Academy  
Use Only

## 演習 9 の解答: データの操作

HR 部門では、従業員データの挿入、更新および削除を行う SQL 文の作成を必要としています。HR 部門に SQL 文を提出する前のプロトタイプとして MY\_EMPLOYEE 表を使用します。

**注意:** すべての DML 文について、「Run Script」アイコンを使用して(または[F5]を押して)問合せを実行します。これにより、「Script Output」タブ・ページにフィードバック・メッセージが表示されます。SELECT 問合せの場合は、さらに「Execute Statement」アイコンを使用するか、[F9]を押すと、「Results」タブ・ページに書式設定された出力が表示されます。

### MY\_EMPLOYEE 表へのデータの挿入

1. lab\_09\_01.sql スクリプトの文を実行して、この演習で使用する MY\_EMPLOYEE 表を作成します。
  - a. 「File」メニューから「Open」を選択します。「Open」ダイアログ・ボックスで、D:\labs\sql\labs フォルダにナビゲートし、lab\_09\_01.sql をダブルクリックします。
  - b. SQL Worksheet で文が開いたら、「Run Script」アイコンをクリックしてスクリプトを実行します。「Script Output」タブ・ページに、表の作成が成功したことを示すメッセージが表示されます。
2. MY\_EMPLOYEE 表の構造を記述して、列の名前を特定します。

```
DESCRIBE my_employee
```

3. 次のサンプルデータから、データの最初の行を MY\_EMPLOYEE 表に追加する INSERT 文を作成します。INSERT 句には列リストを指定しません。

ID	LAST_NAME	FIRST_NAME	USERID	SALARY
1	Patel	Ralph	rpatel	895
2	Dancs	Betty	bdancs	860
3	Biri	Ben	bbiri	1100
4	Newman	Chad	cnewman	750
5	Ropeburn	Audrey	aropebur	1550

```
INSERT INTO my_employee  
VALUES (1, 'Patel', 'Ralph', 'rpatel', 895);
```

## 演習 9 の解答: データの操作(続き)

4. 前述のリストのサンプル・データの 2 行目を MY\_EMPLOYEE 表に移入します。今度は、INSERT 句に列リストを明示的に指定してください。

```
INSERT INTO my_employee (id, last_name, first_name,  
                          userid, salary)  
VALUES (2, 'Dancs', 'Betty', 'bdancs', 860);
```

5. 表への追加結果を確認します。

```
SELECT      *  
FROM        my_employee;
```

6. 残りの行を MY\_EMPLOYEE 表にロードする INSERT 文を、動的に再利用可能なスクリプト・ファイルに記述します。スクリプトでは、すべての列 (ID、LAST\_NAME、FIRST\_NAME、USERID および SALARY) の入力をユーザーに求める必要があります。このスクリプトを lab\_09\_06.sql ファイルに保存します。

```
INSERT INTO my_employee  
VALUES (&p_id, '&p_last_name', '&p_first_name',  
        '&p_userid', &p_salary);
```

7. 作成したスクリプトの INSERT 文を実行して、ステップ 3 で示したサンプル・データの次の 2 行を表に移入します。

```
INSERT INTO my_employee  
VALUES (&p_id, '&p_last_name', '&p_first_name',  
        '&p_userid', &p_salary);
```

8. 表への追加結果を確認します。

```
SELECT      *  
FROM my_employee;
```

9. データの追加を確定します。

```
COMMIT;
```

## 演習 9 の解答: データの操作(続き)

### MY\_EMPLOYEE 表内のデータの更新と削除

10. 従業員 3 の姓を Drexler に変更します。

```
UPDATE my_employee  
SET    last_name = 'Drexler'  
WHERE  id = 3;
```

11. 給与が\$900 未満の従業員すべての給与を\$1,000 に変更します。

```
UPDATE my_employee  
SET    salary = 1000  
WHERE  salary < 900;
```

12. 表の変更結果を確認します。

```
SELECT *  
FROM   my_employee;
```

13. MY\_EMPLOYEE 表から Betty Dancs を削除します。

```
DELETE  
FROM my_employee  
WHERE last_name = 'Dancs';
```

14. 表の変更結果を確認します。

```
SELECT *  
FROM   my_employee;
```

15. 保留中の変更をすべてコミットします。

```
COMMIT;
```

## 演習 9 の解答: データの操作 (続き)

### MY\_EMPLOYEE 表のデータ・トランザクションの制御

16. ステップ 6 で作成したスクリプトの文を使用して、ステップ 3 で示したサンプル・データの最終行を表に移入します。スクリプトの文を実行します。

```
INSERT INTO my_employee  
VALUES (&p_id, '&p_last_name', '&p_first_name',  
       '&p_userid', &p_salary);
```

17. 表への追加結果を確認します。

```
SELECT  *  
FROM    my_employee;
```

18. このトランザクション処理内で、中間点にマーカーを設定します。

```
SAVEPOINT step_17;
```

19. MY\_EMPLOYEE 表からすべての行を削除します。

```
DELETE  
FROM  my_employee;
```

20. 表が空であることを確認します。

```
SELECT  *  
FROM    my_employee;
```

21. 最後の DELETE 操作を破棄します。ただし、それ以前の INSERT 操作は破棄しないでください。

```
ROLLBACK TO step_17;
```

22. 新しい行がそのまま存在することを確認します。

```
SELECT  *  
FROM    my_employee;
```

23. データの追加を確定します。

```
COMMIT;
```

## 演習 9 の解答: データの操作(続き)

時間があるときは、次の演習問題に進みます。

24. lab\_09\_06.sql スクリプトを変更し、名の最初の文字と姓の最初の 7 文字を連結することによって USERID が自動的に生成されるようにします。生成される USERID は必ず小文字にしてください。これで、スクリプトで USERID の入力をユーザーに求める必要はなくなります。このスクリプトを lab\_09\_24.sql というファイルに保存します。

```
SET ECHO OFF
SET VERIFY OFF
INSERT INTO my_employee
VALUES (&p_id, '&p_last_name', '&p_first_name',
       lower(substr('&p_first_name', 1, 1) ||
       substr('&p_last_name', 1, 7)), &p_salary);
SET VERIFY ON
SET ECHO ON
UNDEFINE p_first_name
UNDEFINE p_last_name
```

25. スクリプト lab\_09\_24.sql を実行して、次のレコードを挿入します。

ID	LAST_NAME	FIRST_NAME	USERID	SALARY
6	Anthony	Mark	manthony	1230

26. 新しい行が追加され、正しい USERID が含まれていることを確認します。

```
SELECT *
FROM my_employee
WHERE ID='6';
```

## 演習 10 の解答: DDL 文を使用した表の作成および管理

**注意:** すべての DDL および DML 文について、「Run Script」アイコンを使用して(または[F5]を押して)問合せを実行します。これにより、「Script Output」タブ・ページにフィードバック・メッセージが表示されます。SELECT 問合せの場合は、さらに「Execute Statement」アイコンを使用するか、[F9]を押すと、「Results」タブ・ページに書式設定された出力が表示されます。

1. 次の表インスタンスのチャートに基づいて DEPT 表を作成します。lab\_10\_01.sql というスクリプトに文を保存し、スクリプト内の文を実行して表を作成します。表が作成されたことを確認してください。

```
CREATE TABLE dept
(id    NUMBER(7) CONSTRAINT department_id_pk PRIMARY KEY,
name  VARCHAR2(25));
```

- a. 表が作成されたことを確認し、構造を表示するには、次のコマンドを実行します。

```
DESCRIBE dept
```

2. DEPT 表に DEPARTMENTS 表からデータを移入します。必要な列だけを含めます。

```
INSERT INTO dept
SELECT department_id, department_name
FROM departments;
```

3. 次の表インスタンスのチャートに基づいて EMP 表を作成します。lab\_10\_03.sql というスクリプトに文を保存し、スクリプト内の文を実行して表を作成します。表が作成されたことを確認してください。

```
CREATE TABLE emp
(id          NUMBER(7),
last_name    VARCHAR2(25),
first_name   VARCHAR2(25),
dept_id      NUMBER(7)
CONSTRAINT emp_dept_id_FK REFERENCES dept (id)
);
```

- a. 表が作成されたことを確認し、構造を表示するには、次のコマンドを実行します。

```
DESCRIBE emp
```

4. EMPLOYEES 表の構造に基づいて EMPLOYEES2 表を作成します。EMPLOYEE\_ID、FIRST\_NAME、LAST\_NAME、SALARY および DEPARTMENT\_ID 列のみを含めます。新しい表の列名をそれぞれ ID、FIRST\_NAME、LAST\_NAME、SALARY および DEPT\_ID とします。

```
CREATE TABLE employees2 AS
SELECT employee_id id, first_name, last_name, salary,
       department_id dept_id
FROM employees;
```



## 演習 10 の解答: DDL 文を使用した表の作成および管理(続き)

5. EMPLOYEES2 表の状態を読取り専用に変更します。

```
ALTER TABLE employees2 READ ONLY
```

6. 次の行を EMPLOYEES2 表に挿入してみます。

- a. 表では更新操作ができないことを示すエラー・メッセージが表示されます。したがって、読取り専用状態が割り当てられているため、表に行を挿入することはできません。

```
INSERT INTO employees2  
VALUES (34, 'Grant', 'Marcie', 5678, 10)
```

7. EMPLOYEES2 表を読取り/書込み状態に戻します。ここで、再び同じ行の挿入を試みます。

- a. 表に READ WRITE 状態が割り当てられているため、表に行を挿入することが許可されます。

```
ALTER TABLE employees2 READ WRITE  
  
INSERT INTO employees2  
VALUES (34, 'Grant', 'Marcie', 5678, 10)
```

8. EMPLOYEES2 表を削除します。

- a. **注意:** READ ONLY モードの表を削除することもできます。これをテストするには、表の状態を再度 READ ONLY にして、DROP TABLE コマンドを実行します。表 EMPLOYEES2 が削除されます。

```
DROP TABLE employees2;
```

## 演習 11 の解答: その他のスキーマ・オブジェクトの作成 パート 1

1. HR 部門のスタッフは、EMPLOYEES 表の一部のデータを非表示にすることを必要としています。EMPLOYEES 表の従業員番号、従業員名および部門番号に基づく EMPLOYEES\_VU というビューが必要です。従業員名のヘッダーを EMPLOYEE にしてください。

```
CREATE OR REPLACE VIEW employees_vu AS
  SELECT employee_id, last_name employee, department_id
  FROM employees;
```

2. ビューが機能することを確認します。EMPLOYEES\_VU ビューの内容を表示します。

```
SELECT *
FROM   employees_vu;
```

3. この EMPLOYEES\_VU ビューを使用して、すべての従業員名と部門番号を表示する、HR 部門に対する問合せを記述します。

```
SELECT   employee, department_id
FROM     employees_vu;
```

4. 部門 50 は、従業員データにアクセスする必要があります。部門 50 のすべての従業員の従業員番号、従業員の姓、部門番号が表示される、DEPT50 というビューを作成します。ビューの列には EMPNO、EMPLOYEE および DEPTNO というラベルを付けるように依頼されています。セキュリティ上の理由から、ビューを通じて別の部門に従業員を再割当てする操作を許可しないでください。

```
CREATE VIEW dept50 AS

  SELECT   employee_id empno, last_name employee,
           department_id deptno
  FROM     employees
  WHERE    department_id = 50
  WITH CHECK OPTION CONSTRAINT emp_dept_50;
```

5. DEPT50 ビューの構造とコンテンツを表示します。

```
DESCRIBE dept50

SELECT *
FROM   dept50;
```

## 演習 11 の解答: その他のスキーマ・オブジェクトの作成(続き) パート 2

6. ビューのテストを行います。部門 80 に対して Matos の再割当てを試してみます。

```
UPDATE   dept50
SET      deptno = 80
WHERE    employee = 'Matos';
```

DEPT50 ビューが WITH CHECK OPTION 制約を設定して作成されているため、エラーになります。これにより、ビューの DEPTNO 列が変更されないように保護されます。

7. DEPT 表の PRIMARY KEY 列で利用できる順序が必要です。この順序は 200 から開始し、最大値を 1,000 とします。順序が 10 ずつ増分されるようにします。順序に DEPT\_ID\_SEQ という名前を付けます。

```
CREATE SEQUENCE dept_id_seq
START WITH 200
INCREMENT BY 10
MAXVALUE 1000;
```

8. 順序のテストを行うために、DEPT 表に 2 つの行を挿入するスクリプトを記述します。このスクリプトに lab\_11\_08.sql という名前を付けます。このとき、ID 列に作成した順序を使用してください。2 つの部門 (Education および Administration) を追加します。追加した結果を確認します。スクリプト内のコマンドを実行します。

```
INSERT INTO dept
VALUES (dept_id_seq.nextval, 'Education');

INSERT INTO dept
VALUES (dept_id_seq.nextval, 'Administration');
```

9. DEPT 表の NAME 列に一意ではない索引を作成します。

```
CREATE INDEX dept_name_idx ON dept (name);
```

10. EMPLOYEES 表のシノニムを作成し、EMP という名前を付けます。

```
CREATE SYNONYM emp FOR EMPLOYEES;
```

## 演習 C の解答: Oracle 結合構文

1. HR 部門を検索して、すべての部門の住所を出力する問合せを記述します。LOCATIONS 表と COUNTRIES 表を使用します。出力には、所在地 ID、番地、市、州または県、国を表示します。問合せを実行します。

```
SELECT location_id, street_address, city, state_province, country_name
FROM   locations, countries
WHERE  locations.country_id = countries.country_id;
```

2. HR 部門はすべての従業員のレポートを必要としています。すべての従業員の姓、部門番号および部門名を表示する問合せを記述します。問合せを実行します。

```
SELECT e.last_name, e.department_id, d.department_name
FROM   employees e, departments d
WHERE  e.department_id = d.department_id;
```

3. HR 部門はトロントの従業員のレポートを必要としています。トロントで勤務しているすべての従業員の姓、職務、部門番号および部門名を表示します。

```
SELECT e.last_name, e.job_id, e.department_id, d.department_name
FROM   employees e, departments d, locations l
WHERE  e.department_id = d.department_id
AND    d.location_id = l.location_id
AND    LOWER(l.city) = 'toronto';
```

4. 従業員の姓と従業員番号および担当マネージャの姓とマネージャ番号を表示するレポートを作成します。それぞれの列には Employee、Emp#、Manager および Mgr# のラベルを付けます。作成した SQL 文を lab\_c\_04.sql として保存します。

```
SELECT w.last_name "Employee", w.employee_id "EMP#",
       m.last_name "Manager", m.employee_id  "Mgr#"
FROM   employees w, employees m
WHERE  w.manager_id = m.employee_id;
```

5. 担当マネージャのいない King を含め、すべての従業員が表示されるように lab\_c\_04.sql を変更します。結果は、従業員番号の順に並べます。作成した SQL 文を lab\_c\_05.sql として保存します。lab\_c\_05.sql の問合せを実行します。

```
SELECT w.last_name "Employee", w.employee_id "EMP#",
       m.last_name "Manager", m.employee_id  "Mgr#"
FROM   employees w, employees m
WHERE  w.manager_id = m.employee_id (+);
```

## 演習 C の解答: Oracle 結合構文(続き)

6. HR 部門用のレポートを作成して、従業員の姓、部門番号、および指定した従業員と同じ部門に勤務するすべての従業員を表示します。それぞれの列に、該当するラベルを付けます。作成したスクリプトを lab\_c\_06.sql という名前でファイルに保存します。

```
SELECT e1.department_id department, e1.last_name employee,
       e2.last_name colleague
FROM   employees e1, employees e2
WHERE  e1.department_id = e2.department_id
AND    e1.employee_id <> e2.employee_id
ORDER BY e1.department_id, e1.last_name, e2.last_name;
```

7. HR 部門では、職務等級と給与に関するレポートを必要としています。JOB\_GRADES 表を理解するために、まず、JOB\_GRADES 表の構造を確認します。次に、すべての従業員の名前、職務、部門名、給与および等級を表示する問合せを作成します。

```
DESC JOB_GRADES

SELECT e.last_name, e.job_id, d.department_name,
       e.salary, j.grade_level
FROM   employees e, departments d, job_grades j
WHERE  e.department_id = d.department_id
AND    e.salary BETWEEN j.lowest_sal AND j.highest_sal;
```

さらに演習を続ける場合は、次の演習問題に進みます。

8. HR 部門では、Davies より後に雇用されたすべての従業員の名前を調べています。従業員 Davies より後に雇用されたすべての従業員の名前と雇用日を表示する問合せを作成します。

```
SELECT e.last_name, e.hire_date
FROM   employees e , employees davies
WHERE  davies.last_name = 'Davies'
AND    davies.hire_date < e.hire_date;
```

9. HR 部門では、担当マネージャより前に雇用されたすべての従業員の名前と雇用日、およびその担当マネージャの名前と雇用日を調べる必要があります。列にそれぞれ Employee、Emp Hired、Manager、Mgr Hired というラベルを付けます。作成したスクリプトを lab\_c\_09.sql という名前でファイルに保存します。

```
SELECT w.last_name, w.hire_date, m.last_name, m.hire_date
FROM   employees w , employees m
WHERE  w.manager_id = m.employee_id
AND    w.hire_date < m.hire_date;
```

Oracle Internal & Oracle Academy  
Use Only

# B

## 表の説明

ORACLE

Copyright © 2007, Oracle. All rights reserved.

Oracle Internal & Oracle Academy  
Use Only

## スキーマの説明

### 概要

Oracle Database サンプル・スキーマでは、世界規模で事業を展開し、複数の製品を受注している会社が仮設定されています。この会社には次の3つの部門があります。

- **Human Resources (人事管理)**: 従業員と設備に関する情報を管理します。
- **Order Entry (受注)**: 様々なチャネルを介して製品の在庫と販売を管理します。
- **Sales History (販売履歴)**: ビジネス上の判断に役立つ事業統計を管理します。

この3つの部門は、それぞれスキーマによって表現されます。このコースでは、すべてのスキーマ内のオブジェクトにアクセスできます。ただし、例、デモンストレーションおよび演習では、主に Human Resources (HR) スキーマを使用します。

サンプル・スキーマの作成に必要なスクリプトはすべて \$ORACLE\_HOME/demo/schema/ フォルダにあります。

### Human Resources (HR)

このコースで使用されるスキーマです。Human Resource (HR) レコードには、各従業員の識別番号、電子メール・アドレス、職務識別コード、給与、管理者が含まれています。給与に加えて歩合を受け取る従業員もいます。

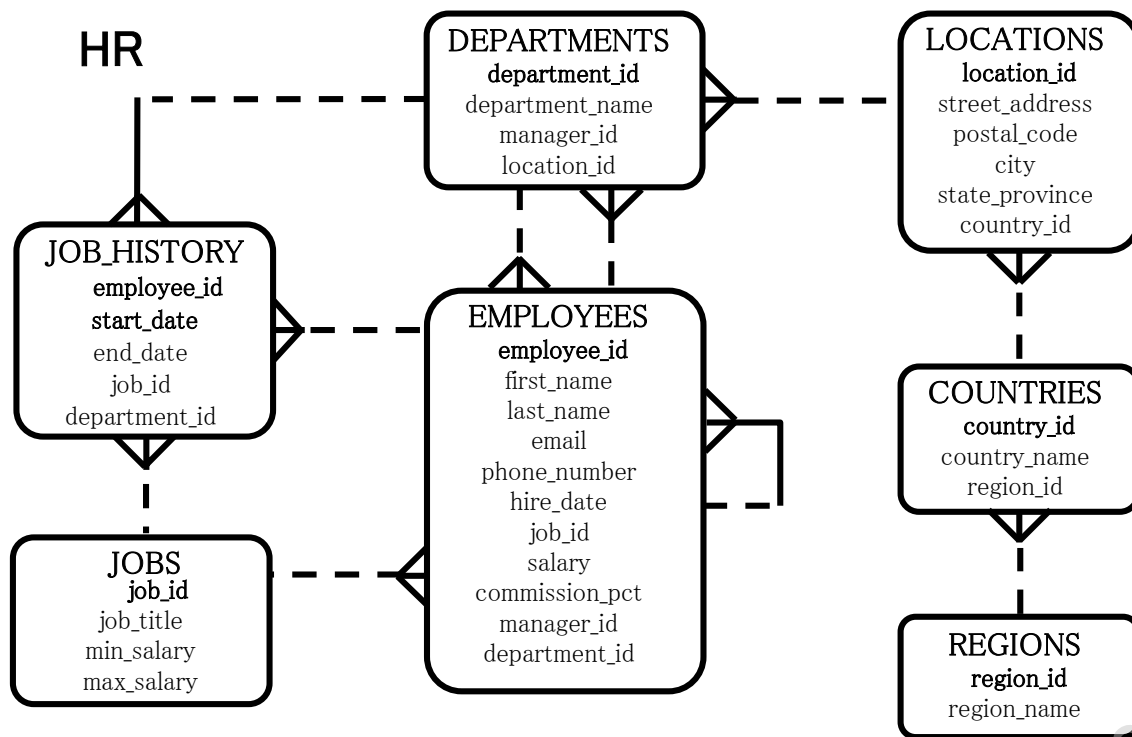
また、会社は組織内の職務についての情報も記録しています。各職務には、識別コード、職務、その職務の給与の上限と下限があります。長期間勤務している従業員の中には、複数の役割を担当している人もいます。従業員が退職すると、その従業員の勤務期間、職務識別番号および部門が記録されます。

サンプルになっている会社は様々な地域に分かれているため、倉庫および部門の所在地を記録しています。各従業員は、部門に配属されており、また各部門は、一意の部門番号または短縮名のいずれかで識別されます。それぞれの部門は1つの場所に関連付けられています。各部門は、1つの所在地に関連付けられており、またそれぞれの所在地には、番地、郵便番号、市、州または県、国コードを含む完全な住所があります。

部門および倉庫の所在地には、国名、通貨記号、通貨名、地理的に位置する地域などの詳細を記録します。



## HRエンティティ関連ダイアグラム



## Human Resources (HR) 表の説明

DESCRIBE countries

Name	Null	Type
-----		
COUNTRY_ID	NOT NULL	CHAR(2)
COUNTRY_NAME		VARCHAR2(40)
REGION_ID		NUMBER

SELECT \* FROM countries;

	COUNTRY_ID	COUNTRY_NAME	REGION_ID
1	CA	Canada	2
2	DE	Germany	1
3	UK	United Kingdom	1
4	US	United States of America	2

Oracle Internal & Oracle Academy  
Use Only

## Human Resources (HR) 表の説明

DESCRIBE departments

Name	Null	Type
DEPARTMENT_ID	NOT NULL	NUMBER(4)
DEPARTMENT_NAME	NOT NULL	VARCHAR2(30)
MANAGER_ID		NUMBER(6)
LOCATION_ID		NUMBER(4)

SELECT \* FROM departments;

	DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
1	10	Administration	200	1700
2	20	Marketing	201	1800
3	50	Shipping	124	1500
4	60	IT	103	1400
5	80	Sales	149	2500
6	90	Executive	100	1700
7	110	Accounting	205	1700
8	190	Contracting	(null)	1700

## Human Resources (HR) 表の説明

DESCRIBE employees

Name	Null	Type
-----	-----	-----
EMPLOYEE_ID	NOT NULL	NUMBER(6)
FIRST_NAME		VARCHAR2(20)
LAST_NAME	NOT NULL	VARCHAR2(25)
EMAIL	NOT NULL	VARCHAR2(25)
PHONE_NUMBER		VARCHAR2(20)
HIRE_DATE	NOT NULL	DATE
JOB_ID	NOT NULL	VARCHAR2(10)
SALARY		NUMBER(8,2)
COMMISSION_PCT		NUMBER(2,2)
MANAGER_ID		NUMBER(6)
DEPARTMENT_ID		NUMBER(4)

SELECT \* FROM employees;

	EMPLOYEE_ID	FIRST_N...	LAST_N...	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	COMMI...	MANAGER_ID	DEPARTMENT_ID
1	100	Steven	King	SKING	515.123.4567	17-JUN-87	AD_PRES	24000	(null)	(null)	90
2	101	Neena	Kochhar	NKOCHHAR	515.123.4568	21-SEP-89	AD_VP	17000	(null)	100	90
3	102	Lex	De Haan	LDEHAAN	515.123.4569	13-JAN-93	AD_VP	17000	(null)	100	90
4	103	Alexander	Hunold	AHUNOLD	590.423.4567	03-JAN-90	IT_PROG	9000	(null)	102	60
5	104	Bruce	Ernst	BERNST	590.423.4568	21-MAY-91	IT_PROG	6000	(null)	103	60
6	107	Diana	Lorentz	DLORENTZ	590.423.5567	07-FEB-99	IT_PROG	4200	(null)	103	60
7	124	Kevin	Mourgos	KMOURGOS	650.123.5234	16-NOV-99	ST_MAN	5800	(null)	100	50
8	141	Trenna	Rajs	TRAJS	650.121.8009	17-OCT-95	ST_CLERK	3500	(null)	124	50
9	142	Curtis	Davies	CDAVIES	650.121.2994	29-JAN-97	ST_CLERK	3100	(null)	124	50
10	143	Randall	Matos	RMATOS	650.121.2874	15-MAR-98	ST_CLERK	2600	(null)	124	50
11	144	Peter	Vargas	PVARGAS	650.121.2004	09-JUL-98	ST_CLERK	2500	(null)	124	50
12	149	Eleni	Zlotkey	EZLOTKEY	011.44.1344.429018	29-JAN-00	SA_MAN	10500	0.2	100	80
13	174	Ellen	Abel	EABEL	011.44.1644.429267	11-MAY-96	SA_REP	11000	0.3	149	80
14	176	Jonathon	Taylor	JTAYLOR	011.44.1644.429265	24-MAR-98	SA_REP	8600	0.2	149	80
15	178	Kimberely	Grant	KGRANT	011.44.1644.429263	24-MAY-99	SA_REP	7000	0.15	149	(null)
16	200	Jennifer	Whalen	JWHALEN	515.123.4444	17-SEP-87	AD_ASST	4400	(null)	101	10
17	201	Michael	Hartstein	MHARTSTE	515.123.5555	17-FEB-96	MK_MAN	13000	(null)	100	20
18	202	Pat	Fay	PFAY	603.123.6666	17-AUG-97	MK_REP	6000	(null)	201	20
19	205	Shelley	Higgins	SHIGGINS	515.123.8080	07-JUN-94	AC_MGR	12000	(null)	101	110
20	206	William	Gietz	WGIEZT	515.123.8181	07-JUN-94	AC_ACC...	8300	(null)	205	110

## Human Resources (HR) 表の説明

DESCRIBE job\_history

DESCRIBE job_history		
Name	Null	Type
-----		
EMPLOYEE_ID	NOT NULL	NUMBER(6)
START_DATE	NOT NULL	DATE
END_DATE	NOT NULL	DATE
JOB_ID	NOT NULL	VARCHAR2(10)
DEPARTMENT_ID		NUMBER(4)

SELECT \* FROM job\_history

	EMPLOYEE_ID	START_DATE	END_DATE	JOB_ID	DEPARTMENT_ID
1	102	13-JAN-93	24-JUL-98	IT_PROG	60
2	101	21-SEP-89	27-OCT-93	AC_ACCOUNT	110
3	101	28-OCT-93	15-MAR-97	AC_MGR	110
4	201	17-FEB-96	19-DEC-99	MK_REP	20
5	114	24-MAR-98	31-DEC-99	ST_CLERK	50
6	122	01-JAN-99	31-DEC-99	ST_CLERK	50
7	200	17-SEP-87	17-JUN-93	AD_ASST	90
8	176	24-MAR-98	31-DEC-98	SA_REP	80
9	176	01-JAN-99	31-DEC-99	SA_MAN	80
10	200	01-JUL-94	31-DEC-98	AC_ACCOUNT	90

## Human Resources (HR) 表の説明

DESCRIBE jobs

Name	Null	Type
-----		
JOB_ID	NOT NULL	VARCHAR2(10)
JOB_TITLE	NOT NULL	VARCHAR2(35)
MIN_SALARY		NUMBER(6)
MAX_SALARY		NUMBER(6)

SELECT \* FROM jobs

	 JOB_ID	 JOB_TITLE	 MIN_SALARY	 MAX_SALARY
1	AD_PRES	President	20000	40000
2	AD_VP	Administration Vice President	15000	30000
3	AD_ASST	Administration Assistant	3000	6000
4	AC_MGR	Accounting Manager	8200	16000
5	AC_ACCOUNT	Public Accountant	4200	9000
6	SA_MAN	Sales Manager	10000	20000
7	SA_REP	Sales Representative	6000	12000
8	ST_MAN	Stock Manager	5500	8500
9	ST_CLERK	Stock Clerk	2000	5000
10	IT_PROG	Programmer	4000	10000
11	MK_MAN	Marketing Manager	9000	15000
12	MK_REP	Marketing Representative	4000	9000

## Human Resources (HR) 表の説明

DESCRIBE locations

Name	Null	Type
-----	-----	-----
LOCATION_ID	NOT NULL	NUMBER(4)
STREET_ADDRESS		VARCHAR2(40)
POSTAL_CODE		VARCHAR2(12)
CITY	NOT NULL	VARCHAR2(30)
STATE_PROVINCE		VARCHAR2(25)
COUNTRY_ID		CHAR(2)

SELECT \* FROM locations

	LOCATION_ID	STREET_ADDRESS	POSTAL_CODE	CITY	STATE_PROVINCE	COUNTRY_ID
1	1400	2014 Jabberwocky Rd	26192	Southlake	Texas	US
2	1500	2011 Interiors Blvd	99236	South San Francisco	California	US
3	1700	2004 Charade Rd	98199	Seattle	Washington	US
4	1800	460 Bloor St. W.	ON M5S 1X8	Toronto	Ontario	CA
5	2500	Magdalen Centre, The Oxford Science Park	OX9 9ZB	Oxford	Oxford	UK

## Human Resources (HR) 表の説明

DESCRIBE regions

Name	Null	Type
-----	-----	-----
REGION_ID	NOT NULL	NUMBER
REGION_NAME		VARCHAR2(25)

SELECT \* FROM regions

	REGION_ID	REGION_NAME
1	1	Europe
2	2	Americas
3	3	Asia
4	4	Middle East and Africa

Oracle Internal & Oracle Academy  
Use Only



# Oracle結合構文

ORACLE

Copyright © 2007, Oracle. All rights reserved.

Oracle Internal & Oracle Academy  
Use Only

## 目的

この付録を終えると、次のことができるようになります。

- 等価結合および非等価結合を使用して複数の表のデータにアクセスするためのSELECT文の記述
- 自己結合による表自体の結合
- 通常は結合条件を満たさないデータの外部結合での表示
- 2つ以上の表のすべての行のデカルト積の生成

ORACLE

Copyright © 2007, Oracle. All rights reserved.

### 目的

この章では、複数の表のデータを取得する方法について説明します。結合は、複数の表の情報を表示するために使用されます。したがって、表を結合することにより、複数の表の情報を表示することができます。

**注意:** 結合については、『Oracle Database SQL言語リファレンス11gリリース1(11.1)』のSQL問合せおよび副問合せの結合に関する節を参照してください。

## 複数の表のデータの取得

**EMPLOYEES**

	EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID
1	100	King	90
2	101	Kochhar	90
3	102	De Haan	90
...			
18	202	Fay	20
19	205	Higgins	110
20	206	Gietz	110

**DEPARTMENTS**

	DEPARTMENT_ID	DEPARTMENT_NAME	LOCATION_ID
1	10	Administration	1700
2	20	Marketing	1800
3	50	Shipping	1500
4	60	IT	1400
5	80	Sales	2500
6	90	Executive	1700
7	110	Accounting	1700
8	190	Contracting	1700

	EMPLOYEE_ID	DEPARTMENT_ID	DEPARTMENT_NAME
1	200	10	Administration
2	201	20	Marketing
3	202	20	Marketing
4	124	50	Shipping
5	144	50	Shipping
...			
18	205	110	Accounting
19	206	110	Accounting

ORACLE

Copyright © 2007, Oracle. All rights reserved.

### 複数の表のデータの取得

複数の表のデータを使用する必要がある場合があります。スライドの例では、2つの個別の表のデータがレポートに表示されています。

- 従業員IDはEMPLOYEES表に表示されています。
- 部門IDはEMPLOYEES表とDEPARTMENTS表の両方に表示されています。
- 部門名はDEPARTMENTS表に表示されています。

レポートを作成するには、EMPLOYEES表とDEPARTMENTS表をリンクし、両方の表のデータにアクセスする必要があります。

## デカルト積

- デカルト積は次の場合に生成されます。
  - 結合条件が省略されている場合
  - 結合条件が無効な場合
  - 1つ目の表のすべての行が2つ目の表のすべての行に結合されている場合
- デカルト積を回避するには、有効な結合条件がWHERE句に常に含まれるようにします。

ORACLE

Copyright © 2007, Oracle. All rights reserved.

### デカルト積

結合条件が無効な場合や完全に省略されている場合、デカルト積が生成され、行のすべての組合せが表示されます。つまり、1つ目の表のすべての行が2つ目の表のすべての行に結合されます。

デカルト積では膨大な数の行が生成されることが多く、その結果はあまり有効ではありません。したがって、すべての表のすべての行を組み合わせる必要が特にないかぎり、必ず有効な結合条件を指定してください。

一方、十分な量のデータをシミュレートするために膨大な数の行を生成する必要があるテストでは、デカルト積が便利です。

# デカルト積の生成

EMPLOYEES (20行)

	EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID
1	100	King	90
2	101	Kochhar	90
3	102	De Haan	90
4	103	Hunold	60
...			
19	205	Higgins	110
20	206	Gietz	110

DEPARTMENTS (8行)

	DEPARTMENT_ID	DEPARTMENT_NAME	LOCATION_ID
1	10	Administration	1700
2	20	Marketing	1800
3	50	Shipping	1500
4	60	IT	1400
5	80	Sales	2500
6	90	Executive	1700
7	110	Accounting	1700
8	190	Contracting	1700

デカルト積:  
20 × 8 = 160行

	EMPLOYEE_ID	DEPARTMENT_ID	LOCATION_ID
1	100	90	1700
2	101	90	1700
3	102	90	1700
4	103	60	1700
...			
159	205	110	1700
160	206	110	1700

ORACLE

Copyright © 2007, Oracle. All rights reserved.

## デカルト積の生成

結合条件が省略されている場合、デカルト積が生成されます。スライドの例では、EMPLOYEES表とDEPARTMENTS表の従業員の姓と部門名が表示されています。結合条件が指定されていないため、EMPLOYEES表のすべての行(20行)がDEPARTMENTS表のすべての行(8行)に結合され、その結果、出力には160の行が生成されます。

```
SELECT last_name, department_name dept_name
FROM employees, departments;
```

	LAST_NAME	DEPT_NAME
1	Abel	Administration
2	Davies	Administration
3	De Haan	Administration
4	Ernst	Administration
5	Fay	Administration
...		
159	Whalen	Contracting
160	Zlotkey	Contracting

## Oracle独自の結合のタイプ

- 等価結合
- 非等価結合
- 外部結合
- 自己結合

ORACLE

Copyright © 2007, Oracle. All rights reserved.

### 結合のタイプ

表の結合には、Oracleの結合構文を使用できます。

**注意:** Oracle9より前のリリースでは、独自の結合構文のみが使用されていました。Oracle独自の結合構文と比べ、SQL:1999準拠の結合構文にはパフォーマンス上の利点はありません。

Oracleには、SQL:1999準拠の結合構文のFULL OUTER JOINをサポートする構文はありません。

## Oracle構文を使用した表の結合

複数の表のデータを問い合わせるには、次のように結合を使用します。

```
SELECT    table1.column, table2.column
FROM      table1, table2
WHERE     table1.column1 = table2.column2;
```

WHERE句に結合条件を記述します。

- 複数の表に同じ列名が表示される場合は、列名の前に表名を付けます。

ORACLE

Copyright © 2007, Oracle. All rights reserved.

### Oracle構文を使用した表の結合

データベース内にある複数の表のデータが必要な場合は、結合条件を使用します。対応する列（通常は主キー列と外部キー列）に存在する共通の値に従って、1つの表内の行を別の表内の行に結合できます。

関連する2つ以上の表のデータを表示するには、WHERE句に単純な結合条件を記述します。

構文の内容:

*table1.column*

データの取得元の表と列を示します。

*table1.column1 =*

表を結合する(または関連付ける)ための条件です。

*table2.column2*

#### ガイドライン

- SELECT文を記述して表を結合する場合は、列を明確にするために列名の前に表名を付けて、データベース・アクセスを向上させます。
- 複数の表に同じ列名が表示される場合は、列名の前に表名を付ける必要があります。
- $n$ 個の表を結合するには、少なくとも $n-1$ の結合条件が必要です。たとえば、4つの表を結合するには、少なくとも3つの結合が必要です。表に連結主キーがある場合は、この規則は適用されません。この場合は、各行を一意に識別するために複数の列が必要になります。

## あいまいな列名の修飾

- 複数の表にある列名を修飾するには、表接頭辞を使用します。
- 表接頭辞を使用すると、パフォーマンスが向上します。
- 完全な名前の表接頭辞ではなく、表別名を使用します。
- 表別名は表の短い名前です。
  - SQLコードが短くなるため、メモリーの使用量が削減される
- 異なる表にある同一名の列を区別するには、列別名を使用します。

ORACLE

Copyright © 2007, Oracle. All rights reserved.

### あいまいな列名の修飾

2つ以上の表を結合する場合、表の名前で列名を修飾して、あいまいさを回避する必要があります。表接頭辞を使用しない場合、SELECTリストのDEPARTMENT\_ID列は、DEPARTMENTS表またはEMPLOYEES表のいずれからでも取得できます。したがって、表接頭辞を追加して問合せを実行する必要があります。2つの表に共通の列名がない場合には、列の修飾は不要です。ただし、表接頭辞を使用すると、検索する列の正確な場所がOracleサーバーによって認識されるため、パフォーマンスが向上します。

表の名前による列名の修飾には時間がかかることがあります。表名が長い場合には特に時間がかかります。そこで、表名を使用する代わりに、表別名を使用します。列別名が列の別名であるのと同様に、表別名は表の別名です。表別名を使用することにより、SQLコードが短くなるため、メモリーの使用量を削減できます。

表の名前には、完全な名前を指定し、その後に空白と表別名を続けます。たとえば、EMPLOYEES表には別名eを指定し、DEPARTMENTS表には別名dを指定できます。

#### ガイドライン

- 表別名には30文字まで指定できますが、別名は短い方が便利です。
- FROM句で特定の表名に対して表別名を使用した場合、SELECT文では表名の代わりにその表別名を使用する必要があります。
- 表別名には、わかりやすい名前を指定する必要があります。
- 表別名は、現在のSELECT文に対してのみ有効です。



## 等価結合

EMPLOYEES

EMPLOYEE_ID	DEPARTMENT_ID
100	90
101	90
102	90
103	60
104	60
107	60
124	50
141	50
142	50
143	50
144	50
149	80
174	80
176	80
...	...

DEPARTMENTS

DEPARTMENT_ID	DEPARTMENT_NAME
1	10 Administration
2	20 Marketing
3	50 Shipping
4	60 IT
5	80 Sales
6	90 Executive
7	110 Accounting
8	190 Contracting

主キー

外部キー

ORACLE

Copyright © 2007, Oracle. All rights reserved.

### 等価結合

従業員の部門名を確認するには、EMPLOYEES表のDEPARTMENT\_ID列の値とDEPARTMENTS表のDEPARTMENT\_IDの値を比較します。EMPLOYEES表とDEPARTMENTS表の関係は等価結合です。つまり、両方の表のDEPARTMENT\_ID列に同じ値が使用されている必要があります。通常、このタイプの結合には、主キーと外部キーの補集合が使用されます。

**注意:** 等価結合は、単純結合または内部結合とも呼ばれます。

## 等価結合によるレコードの取得

```
SELECT e.employee_id, e.last_name, e.department_id,  
       d.department_id, d.location_id  
FROM   employees e, departments d  
WHERE  e.department_id = d.department_id;
```

	EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID	DEPARTMENT_ID_1	LOCATION_ID
1	200	Whalen	10	10	1700
2	201	Hartstein	20	20	1800
3	202	Fay	20	20	1800
4	124	Mourgos	50	50	1500
5	144	Vargas	50	50	1500
6	143	Matos	50	50	1500
7	142	Davies	50	50	1500
8	141	Rajs	50	50	1500
9	107	Lorentz	60	60	1400
10	104	Ernst	60	60	1400

...

ORACLE

Copyright © 2007, Oracle. All rights reserved.

### 等価結合によるレコードの取得

スライドの例では次のように指定されています。

- SELECT句によって、取得する列名が次のように指定されています。
  - EMPLOYEES表の列に表示されている従業員の姓、従業員番号および部門番号
  - DEPARTMENTS表の列に表示されている部門番号、部門名および所在地ID
- FROM句によって、データベースでアクセスする必要がある表として、次の2つの表が指定されています。
  - EMPLOYEES表
  - DEPARTMENTS表
- WHERE句によって、表を結合する方法が次のように指定されます。

e.department\_id = d.department\_id

DEPARTMENT\_ID列は両方の表に共通しているため、この列の前に表の別名を付けて、あいまいさを回避する必要があります。片方の表にしかない他の列を表の別名で修飾する必要はありませんが、パフォーマンスを向上させるため修飾することをお勧めします。

注意: SQL Developerでは、\_1を接尾辞に使用して、2つのDEPARTMENT\_IDを区別します。

## 等価結合によるレコードの取得: 例

```
SELECT d.department_id, d.department_name,  
       d.location_id, l.city  
FROM   departments d, locations l  
WHERE  d.location_id = l.location_id;
```

	DEPARTMENT_ID	DEPARTMENT_NAME	LOCATION_ID	CITY
1	60	IT	1400	Southlake
2	50	Shipping	1500	South San Francisco
3	10	Administration	1700	Seattle
4	90	Executive	1700	Seattle
5	110	Accounting	1700	Seattle
6	190	Contracting	1700	Seattle
7	20	Marketing	1800	Toronto
8	80	Sales	2500	Oxford

ORACLE

Copyright © 2007, Oracle. All rights reserved.

### 等価結合によるレコードの取得: 例

スライドの例では、LOCATIONS表は、LOCATION\_ID列によってDEPARTMENTS表に結合されています。この列は、両方の表で同じ名前を持つ唯一の列です。表別名を使用して列を修飾すると、あいまいさを回避できます。

## AND演算子を使用した追加検索条件

```
SELECT  d.department_id, d.department_name, l.city
FROM    departments d, locations l
WHERE   d.location_id = l.location_id
AND d.department_id IN (20, 50);
```

	DEPARTMENT_ID	DEPARTMENT_NAME	CITY
1	20	Marketing	Toronto
2	50	Shipping	South San Francisco

ORACLE

Copyright © 2007, Oracle. All rights reserved.

### AND演算子を使用した追加検索条件

結合に加え、WHERE句に条件を指定して、1つ以上の表に対して結合の対象となる行を制限できます。スライドの例では、出力される行は、部門IDに20または50が指定されている行に制限されています。

たとえば、従業員Matosの部門番号と部門名を表示するには、WHERE句に追加条件が必要です。

```
SELECT  e.last_name, e.department_id,
        d.department_name
FROM    employees e, departments d
WHERE   e.department_id = d.department_id
AND     last_name = 'Matos';
```

	LAST_NAME	DEPARTMENT_ID	DEPARTMENT_NAME
1	Matos	50	Shipping

## 3つ以上の表の結合

EMPLOYEES

	A2	LAST_NAME	A2	DEPARTMENT_ID
1		King		90
2		Kochhar		90
3		De Haan		90
4		Hunold		60
5		Ernst		60
6		Lorentz		60
7		Mourgos		50
8		Rajs		50
9		Davies		50
10		Matos		50

...

DEPARTMENTS

	A2	DEPARTMENT_ID	A2	LOCATION_ID
		10		1700
		20		1800
		50		1500
		60		1400
		80		2500
		90		1700
		110		1700
		190		1700

LOCATIONS

	A2	LOCATION_ID	A2	CITY
		1400		Southlake
		1500		South San Francisco
		1700		Seattle
		1800		Toronto
		2500		Oxford

$n$ 個の表を結合するには、少なくとも $n-1$ 個の結合条件が必要です。たとえば、3つの表を結合するには、少なくとも2つの結合が必要です。

ORACLE

Copyright © 2007, Oracle. All rights reserved.

### 3つ以上の表の結合

3つ以上の表の結合が必要になる場合があります。たとえば、各従業員の姓、部門名および市を表示するには、EMPLOYEES、DEPARTMENTSおよびLOCATIONS表を結合する必要があります。

```
SELECT e.last_name, d.department_name, l.city
FROM   employees e, departments d, locations l
WHERE  e.department_id = d.department_id
AND    d.location_id = l.location_id;
```

	A2	LAST_NAME	A2	DEPARTMENT_NAME	A2	CITY
1		Abel		Sales		Oxford
2		Davies		Shipping		South San Francisco
3		De Haan		Executive		Seattle
4		Ernst		IT		Southlake
5		Fay		Marketing		Toronto

...

## 非等価結合

EMPLOYEES

EMPLOYEE_ID	LAST_NAME	SALARY
1	King	24000
2	Kochhar	17000
3	De Haan	17000
4	Hunold	9000
5	Ernst	6000
6	Lorentz	4200
7	Mourgos	5800
8	Rajs	3500
9	Davies	3100
10	Matos	2600
...		
19	Higgins	12000
20	Gietz	8300

JOB\_GRADES

GRADE_LEVEL	LOWEST_SAL	HIGHEST_SAL
1 A	1000	2999
2 B	3000	5999
3 C	6000	9999
4 D	10000	14999
5 E	15000	24999
6 F	25000	40000

JOB\_GRADES表によって、それぞれのGRADE\_LEVELに対応するLOWEST\_SALとHIGHEST\_SALの値の範囲が定義されます。したがって、GRADE\_LEVEL列を使用して、それぞれの従業員に等級を割り当てることができます。

ORACLE

Copyright © 2007, Oracle. All rights reserved.

### 非等価結合

非等価結合は、等価演算子ではない演算子による結合条件です。

EMPLOYEES表とJOB\_GRADES表の関係は、非等価結合となります。EMPLOYEES表のSALARY列の値は、JOB\_GRADES表のLOWEST\_SAL列とHIGHEST\_SAL列の値の範囲内にあります。したがって、各従業員を給与に基づいて等級分けすることができます。この関係の取得には、等価演算子(=)ではない演算子が使用されます。

## 非等価結合によるレコードの取得

```
SELECT e.last_name, e.salary, j.grade_level
FROM   employees e, job_grades j
WHERE  e.salary
       BETWEEN j.lowest_sal AND j.highest_sal;
```

	LAST_NAME	SALARY	GRADE_LEVEL
1	Vargas	2500	A
2	Matos	2600	A
3	Davies	3100	B
4	Rajs	3500	B
5	Lorentz	4200	B
6	Whalen	4400	B
7	Mourgos	5800	B
8	Ernst	6000	C
9	Fay	6000	C
10	Grant	7000	C

...

ORACLE

Copyright © 2007, Oracle. All rights reserved.

### 非等価結合によるレコードの取得

スライドの例では、従業員の給与等級を評価するために非等価結合が作成されています。給与は、下限と上限のいずれかの組合せの範囲内に必要があります。

この問合せを実行したときに、すべての従業員が一度だけしか表示されないことに注意してください。従業員が重複してリストに表示されることはありません。これには、次の2つの理由があります。

- 職務等級表には、重複している等級の行はありません。すなわち、従業員の給与の値は、給与等級表のいずれか1つの行の上限と下限の組合せの範囲内にあります。
- すべての従業員の給与は、職務等級表で設定されている範囲内にあります。すなわち、給与がLOWEST\_SAL列の最低値よりも低い従業員や、HIGHEST\_SAL列の最高値を超える従業員は存在しません。

**注意:** 別の条件(<= や >= など)を使用することもできますが、BETWEENは最も簡単に使用できます。BETWEEN条件を使用する際には、最初に下限値を指定し、次に上限値を指定してください。Oracleサーバーでは、BETWEEN条件はAND条件の組合せに変換されます。したがって、BETWEEN条件を使用してもパフォーマンス上の利点はありませんが、論理的に使用できるため操作は簡単です。

スライドの例では、あいまいさを回避するためではなく、パフォーマンス上の理由から表別名が指定されています。

## 外部結合による 直接一致しないレコードの取得

DEPARTMENT

DEPARTMENT_NAME	DEPARTMENT_ID
Administration	10
Marketing	20
Shipping	50
IT	60
Sales	80
Executive	90
Accounting	110
Contracting	190

EMPLOYEES

DEPARTMENT_ID	LAST_NAME
1	90 King
2	90 Kochhar
3	90 De Haan
4	60 Hunold
5	60 Ernst
6	60 Lorentz
7	50 Mourgos
8	50 Rajs
9	50 Davies
10	50 Matos
...	
19	110 Higgins
20	110 Gietz

部門190には従業員が存在しない。

ORACLE

Copyright © 2007, Oracle. All rights reserved.

### 外部結合による直接一致しないレコードの取得

行が結合条件を満たしていない場合、その行は問合せ結果に表示されません。たとえば、EMPLOYEES表とDEPARTMENTS表の等価結合条件では、部門ID190の従業員はEMPLOYEES表に存在しないため、この部門IDは表示されません。同じように、DEPARTMENT\_IDがNULLに設定されている従業員が存在する場合は、この行も等価結合による問合せ結果には表示されません。従業員が存在しない部門レコードまたは部門に所属していない従業員レコードを戻すには、外部結合を使用します。



## 外部結合: 構文

- 外部結合を使用して、結合条件に一致しない行を表示します。
- 外部結合の演算子はプラス記号(+)です。

```
SELECT table1.column, table2.column
FROM   table1, table2
WHERE  table1.column(+) = table2.column;
```

```
SELECT table1.column, table2.column
FROM   table1, table2
WHERE  table1.column = table2.column(+);
```

ORACLE

Copyright © 2007, Oracle. All rights reserved.

### 外部結合: 構文

結合条件で外部結合の演算子を使用すると、欠落している行を戻すことができます。演算子にはカッコで囲んだプラス記号(+)を使用し、情報が不足している側の結合に指定します。この演算子を使用すると、NULL行が作成され、情報が不足していない表の行がこのNULL行に結合されます。

構文の内容:

*table1.column* = 表を結合する(または関連付ける)条件です。

*table2.column* (+) WHERE句条件のいずれかの側に指定できる(両側には指定できない)外部結合の記号です(一致する行がない表の列名の後に外部結合の記号を指定します)。

## 外部結合の使用

```
SELECT e.last_name, e.department_id, d.department_name
FROM   employees e, departments d
WHERE  e.department_id(+) = d.department_id ;
```

R2	LAST_NAME	R2	DEPARTMENT_ID	R2	DEPARTMENT_NAME
1	Whalen		10		Administration
2	Hartstein		20		Marketing
3	Fay		20		Marketing
4	Davies		50		Shipping
5	Vargas		50		Shipping
6	Rajs		50		Shipping
7	Mourgos		50		Shipping
8	Matos		50		Shipping
9	Hunold		60		IT
10	Ernst		60		IT
...					
19	Gietz		110		Accounting
20	(null)		(null)		Contracting

ORACLE

Copyright © 2007, Oracle. All rights reserved.

### 外部結合の使用

スライドの例では、従業員の姓、部門IDおよび部門名が表示されています。契約部門には、従業員が存在しません。出力には空の値が表示されます。

### 外部結合の制限

- 外部結合の演算子は、一方の側(情報が欠落している側)の式にのみ指定できます。この演算子は、1つの表から、他の表で直接一致しない行を戻します。
- 外部結合に関連する条件には、IN演算子を使用できません。また、その条件をOR演算子で別の条件にリンクすることもできません。

**注意:** Oracleの結合構文には、SQL:1999準拠の結合構文のFULL OUTER JOINに相当する構文はありません。

## 外部結合: 別の例

```
SELECT e.last_name, e.department_id, d.department_name
FROM   employees e, departments d
WHERE  e.department_id = d.department_id(+);
```

	LAST_NAME	DEPARTMENT_ID	DEPARTMENT_NAME
1	Whalen	10	Administration
2	Fay	20	Marketing
3	Hartstein	20	Marketing
4	Vargas	50	Shipping
5	Matos	50	Shipping
...			
17	King	90	Executive
18	Gietz	110	Accounting
19	Higgins	110	Accounting
20	Grant	(null)	(null)

ORACLE

Copyright © 2007, Oracle. All rights reserved.

### 外部結合: 別の例

このスライドの例の間合せでは、DEPARTMENTS表で一致するものがない場合でも、EMPLOYEES表内のすべての行が取得されます。

## 表自体の結合

EMPLOYEES (WORKER)

	EMPLOYEE_ID	LAST_NAME	MANAGER_ID
1	100	King	(null)
2	101	Kochhar	100
3	102	De Haan	100
4	103	Hunold	102
5	104	Ernst	103
6	107	Lorentz	103
7	124	Mourgos	100
8	141	Rajs	124
9	142	Davies	124
10	143	Matos	124
...			

EMPLOYEES (MANAGER)

EMPLOYEE_ID	LAST_NAME
100	King
101	Kochhar
102	De Haan
103	Hunold
104	Ernst
107	Lorentz
124	Mourgos
141	Rajs
142	Davies
143	Matos
...	

WORKER表のMANAGER\_IDと、  
MANAGER表のEMPLOYEE\_IDは同じである。

ORACLE

Copyright © 2007, Oracle. All rights reserved.

### 表自体の結合

表をその表自体に結合する必要がある場合があります。各従業員のマネージャの名前を検索するには、EMPLOYEES表をその表自体に結合する(すなわち、自己結合を実行する)必要があります。たとえば、Lorentzのマネージャの名前を検索するには、次の手順を実行する必要があります。

- EMPLOYEES表のLAST\_NAME列でLorentzを検索します。
- MANAGER\_ID列でLorentzのマネージャ番号を確認します。Lorentzのマネージャ番号は103です。
- LAST\_NAME列でEMPLOYEE\_IDが103のマネージャの名前を検索します。Hunoldの従業員番号が103です。したがって、HunoldがLorentzのマネージャとなります。

この手順では、表を2回検索します。1回目は、LAST\_NAME列でLorentzを検索し、MANAGER\_IDの値(103)を確認しています。2回目は、EMPLOYEE\_ID列で103を検索し、LAST\_NAME列でHunoldを確認しています。

## 自己結合: 例

```
SELECT worker.last_name || ' works for '
       || manager.last_name
FROM   employees worker, employees manager
WHERE  worker.manager_id = manager.employee_id ;
```

	WORKER.LAST_NAME  'WORKS FOR'  MANAGER.LAST_NAME
1	Hunold works for De Haan
2	Fay works for Hartstein
3	Gietz works for Higgins
4	Lorentz works for Hunold
5	Ernst works for Hunold
6	Zlotkey works for King
7	Mourgos works for King
8	Kochhar works for King
9	Hartstein works for King
10	De Haan works for King

...

ORACLE

Copyright © 2007, Oracle. All rights reserved.

### 自己結合: 例

スライドの例では、EMPLOYEES表がその表自体に結合されています。FROM句で2つの表をシミュレートするため、2つの別名 (workerとmanager) が、同じ表のEMPLOYEESに使用されています。この例では、WHERE句に、workerのマネージャ番号がそのマネージャの従業員番号に一致することを表す結合が含まれています。

## まとめ

この付録では、Oracle独自の構文を使用して複数の表のデータを表示する結合方法を学習しました。

ORACLE

Copyright © 2007, Oracle. All rights reserved.

### まとめ

表の結合には複数の方法があります。

#### 結合のタイプ

- デカルト積
- 等価結合
- 非等価結合
- 外部結合
- 自己結合

#### デカルト積

デカルト積では、行のすべての組合せが表示されます。これは、WHERE句を省略すると実行されます。

#### 表別名

- 表別名によりデータベースのアクセス速度が向上します。
- 表別名を使用すると、SQLコードを短くし、メモリーを節約することができます。

## 演習C: 概要

この演習では次の項目について説明しています。

- 等価結合を使用した表の結合
- 外部結合および自己結合の実行
- 条件の追加

ORACLE

Copyright © 2007, Oracle. All rights reserved.

### 演習C: 概要

この演習では、実際にOracle結合構文を使用して複数の表からデータを抽出します。

## 演習C

1. HR部門を検索して、すべての部門の住所を出力する問合せを記述します。LOCATIONS表とCOUNTRIES表を使用します。出力には、所在地ID、番地、市、州または県、国を表示します。問合せを実行します。

	LOCATION_ID	STREET_ADDRESS	CITY	STATE_PROVINCE	COUNTRY_NAME
1	1400	2014 Jabberwocky Rd	Southlake	Texas	United States of America
2	1500	2011 Interiors Blvd	South San Francisco	California	United States of America
3	1700	2004 Charade Rd	Seattle	Washington	United States of America
4	1800	460 Bloor St. W.	Toronto	Ontario	Canada
5	2500	Magdalen Centre, The ...	Oxford	Oxford	United Kingdom





2. HR部門はすべての従業員のレポートを必要としています。すべての従業員の姓、部門番号および部門名を表示する問合せを記述します。問合せを実行します。

	LAST_NAME	DEPARTMENT_ID	DEPARTMENT_NAME
1	Whalen	10	Administration
2	Hartstein	20	Marketing
3	Fay	20	Marketing
4	Davies	50	Shipping
5	Vargas	50	Shipping
6	Rajs	50	Shipping
7	Mourgos	50	Shipping
8	Matos	50	Shipping
9	Hunold	60	IT
10	Ernst	60	IT
...			
18	Higgins	110	Accounting
19	Gietz	110	Accounting



## 演習C(続き)

- HR部門はトロントの従業員のレポートを必要としています。トロントで勤務しているすべての従業員の姓、職務、部門番号および部門名を表示します。

	 LAST_NAME	 JOB_ID	 DEPARTMENT_ID	 DEPARTMENT_NAME
1	Hartstein	MK_MAN	20	Marketing
2	Fay	MK_REP	20	Marketing

- 従業員の姓と従業員番号および担当マネージャの姓とマネージャ番号を表示するレポートを作成します。それぞれの列にはEmployee、Emp#、ManagerおよびMgr#のラベルを付けます。作成したSQL文をlab\_c\_04.sqlとして保存します。

	 Employee	 EMP#	 Manager	 Mgr#
1	Kochhar	101	King	100
2	De Haan	102	King	100
3	Hunold	103	De Haan	102
4	Ernst	104	Hunold	103
5	Lorentz	107	Hunold	103
6	Mourgos	124	King	100
7	Rajs	141	Mourgos	124
8	Davies	142	Mourgos	124
9	Matos	143	Mourgos	124
10	Vargas	144	Mourgos	124

...

15	Whalen	200	Kochhar	101
16	Hartstein	201	King	100
17	Fay	202	Hartstein	201
18	Higgins	205	Kochhar	101
19	Gietz	206	Higgins	205

## 演習C(続き)

- 担当マネージャのいないKingを含め、すべての従業員が表示されるようにlab\_c\_04.sqlを変更します。結果は、従業員番号の順に並べます。作成したSQL文をlab\_c\_05.sqlとして保存します。lab\_c\_05.sqlの問合せを実行します。

	Employee	EMP#	Manager	Mgr#
1	Hunold	103	De Haan	102
2	Fay	202	Hartstein	201
3	Gietz	206	Higgins	205
4	Lorentz	107	Hunold	103
5	Ernst	104	Hunold	103
6	Hartstein	201	King	100
7	Zlotkey	149	King	100
8	Mourgos	124	King	100
9	De Haan	102	King	100
10	Kochhar	101	King	100

...

17	Grant	178	Zlotkey	149
18	Taylor	176	Zlotkey	149
19	Abel	174	Zlotkey	149
20	King	100 (null)		(null)

- HR部門用のレポートを作成して、従業員の姓、部門番号、および指定した従業員と同じ部門に勤務するすべての従業員を表示します。それぞれの列に、該当するラベルを付けます。作成したスクリプトをlab\_c\_06.sqlという名前でファイルに保存します。

	DEPARTMENT	EMPLOYEE	COLLEAGUE
1	20	Fay	Hartstein
2	20	Hartstein	Fay
3	50	Davies	Matos
4	50	Davies	Mourgos
5	50	Davies	Rajs
6	50	Davies	Vargas
7	50	Matos	Davies
8	50	Matos	Mourgos
9	50	Matos	Rajs
10	50	Matos	Vargas

...

42	110	Higgins	Gietz
----	-----	---------	-------

## 演習C(続き)

- HR部門では、職務等級と給与に関するレポートを必要としています。JOB\_GRADES表を理解するために、まず、JOB\_GRADES表の構造を確認します。次に、すべての従業員の名前、職務、部門名、給与および等級を表示する問合せを作成します。

```
DESC JOB_GRADES
```

Name	Null	Type
GRADE_LEVEL		VARCHAR2(3)
LOWEST_SAL		NUMBER
HIGHEST_SAL		NUMBER


3 rows selected

	LAST_NAME	JOB_ID	DEPARTMENT_NAME	SALARY	GRADE_LEVEL
1	Vargas	ST_CLERK	Shipping	2500	A
2	Matos	ST_CLERK	Shipping	2600	A
3	Davies	ST_CLERK	Shipping	3100	B
4	Rajs	ST_CLERK	Shipping	3500	B
5	Lorentz	IT_PROG	IT	4200	B
6	Whalen	AD_ASST	Administration	4400	B
7	Mourgos	ST_MAN	Shipping	5800	B
8	Ernst	IT_PROG	IT	6000	C
9	Fay	MK_REP	Marketing	6000	C
10	Gietz	AC_ACCOUNT	Accounting	8300	C
...					
18	De Haan	AD_VP	Executive	17000	E
19	King	AD PRES	Executive	24000	E


## 演習C(続き)

さらに演習を続ける場合は、次の演習問題に進みます。

- HR部門では、Daviesより後に雇用されたすべての従業員の名前を調べています。従業員Daviesより後に雇用されたすべての従業員の名前と雇用日を表示する問合せを作成します。

	 LAST_NAME	HIRE_DATE
1	Lorentz	07-FEB-99
2	Mourgos	16-NOV-99
3	Matos	15-MAR-98
4	Vargas	09-JUL-98
5	Zlotkey	29-JAN-00
6	Taylor	24-MAR-98
7	Grant	24-MAY-99
8	Fay	17-AUG-97

- HR部門では、担当マネージャより前に雇用されたすべての従業員の名前と雇用日、およびその担当マネージャの名前と雇用日を調べる必要があります。作成したスクリプトを lab\_c\_09.sql という名前でファイルに保存します。

	 LAST_NAME	HIRE_DATE	 LAST_NAME_1	HIRE_DATE_1
1	Whalen	17-SEP-87	Kochhar	21-SEP-89
2	Hunold	03-JAN-90	De Haan	13-JAN-93
3	Vargas	09-JUL-98	Mourgos	16-NOV-99
4	Matos	15-MAR-98	Mourgos	16-NOV-99
5	Davies	29-JAN-97	Mourgos	16-NOV-99
6	Rajs	17-OCT-95	Mourgos	16-NOV-99
7	Grant	24-MAY-99	Zlotkey	29-JAN-00
8	Taylor	24-MAR-98	Zlotkey	29-JAN-00
9	Abel	11-MAY-96	Zlotkey	29-JAN-00



# SQL\*Plusの使用

ORACLE

Copyright © 2007, Oracle. All rights reserved.

Oracle Internal & Oracle Academy  
Use Only

## 目的

この付録を終えると、次のことができるようになります。

- SQL\*Plusへのログイン
- SQLコマンドの編集
- SQL\*Plusコマンドを使用した出力の書式設定
- スクリプト・ファイルとの相互作用

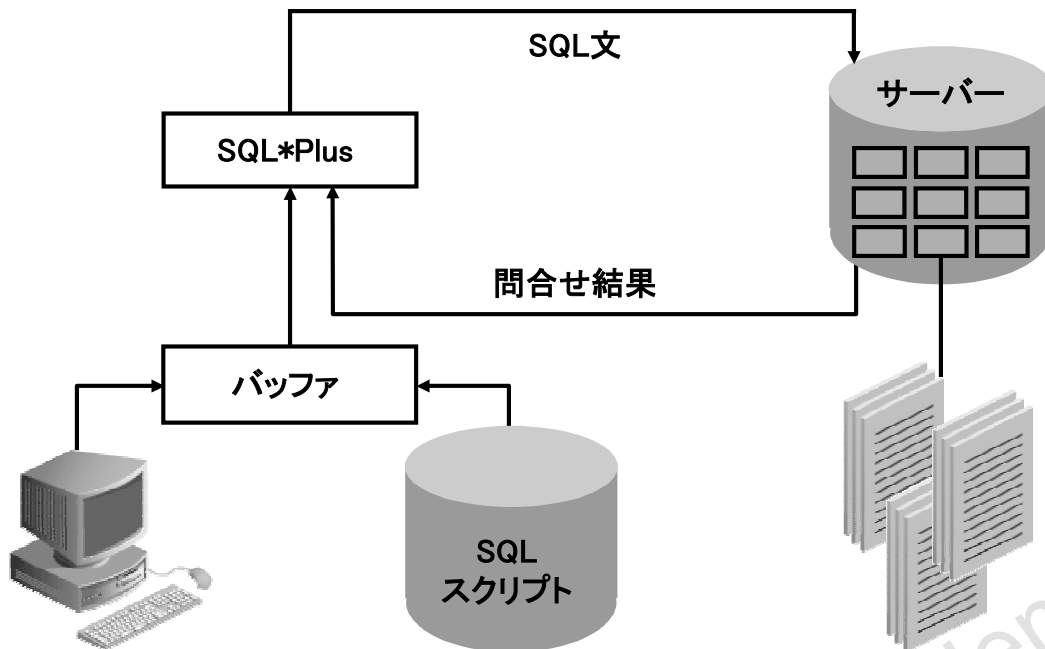
ORACLE

Copyright © 2007, Oracle. All rights reserved.

### 目的

繰り返し使用できるSELECT文の作成が必要になる場合があります。この付録では、SQL\*Plusコマンドを使用したSQL文の実行についても説明しています。SQL\*Plusコマンドによる出力の書式設定、SQLコマンドの編集、およびSQL\*Plusでのスクリプトの保存方法について学習します。

## SQLとSQL\*Plusの相互作用



Copyright © 2007, Oracle. All rights reserved.

### SQLとSQL\*Plus

SQLは、任意のツールまたはアプリケーションとOracleサーバーとの通信に使用されるコマンド言語です。Oracle SQLには、多くの拡張機能が含まれています。SQL文を入力すると、そのSQL文はSQLバッファと呼ばれるメモリーの一部に格納されて、新しいSQL文を入力するまでバッファ内に残ります。SQL\*Plusは、SQL文を実行するために、SQL文を認識してOracle9i ServerにサブミットするOracleツールです。SQL\*Plusには、独自のコマンド言語があります。

#### SQLの特徴

- プログラミングの経験がほとんどまたはまったくないユーザーも含め、様々なユーザーが使用できます。
- 非手続き型言語です。
- システムの作成と保守に必要な時間を軽減します。
- 英語に似た言語です。

#### SQL\*Plusの特徴

- 文の非定型エントリを受け入れます。
- ファイルからのSQL入力を受け入れます。
- SQL文を変更できるライン・エディタを提供します。
- 環境設定を制御します。
- 問合せ結果を基本レポートの書式に設定します。
- ローカルおよびリモート・データベースにアクセスします。

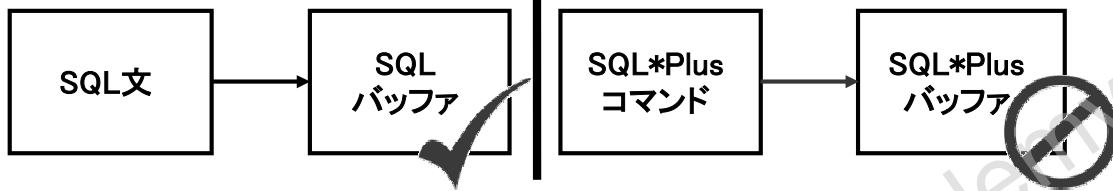
# SQL文とSQL\*Plusコマンド

## SQL

- 言語
- ANSI標準
- キーワードを省略できない
- 文によってデータベース内のデータおよび表定義が操作される

## SQL\*Plus

- 環境
- Oracle独自
- キーワードを省略できる
- コマンドでデータベース内の値を操作できない



ORACLE

Copyright © 2007, Oracle. All rights reserved.

## SQLとSQL\*Plus(続き)

次の表に、SQLとSQL\*Plusの比較を示します。

SQL	SQL*Plus
Oracleサーバーと通信してデータにアクセスするための言語です。	SQL文を認識してサーバーに送信します。
米国規格協会(ANSI)標準SQLに基づきます。	SQL文を実行するためのOracle独自のインタフェースです。
データベース内のデータおよび表定義を操作します。	データベース内の値を操作できません。
1つ以上の行でSQLバッファに入力されます。	1回に1つの行で入力されます。SQLバッファには格納されません。
継続文字は使用しません。	コマンドが1行を超える場合は、ダッシュ(-)を継続文字として使用します。
省略形式を使用できません。	省略形式を使用できます。
終了文字を使用してコマンドを即時に実行します。	終了文字は必要ありません。コマンドを即時に実行します。
関数を使用して一部の書式設定を行います。	コマンドを使用して、データを書式設定します。



# SQL\*Plusの概要

- SQL\*Plusへのログイン
- 表構造の説明
- SQL文の編集
- SQL\*PlusからのSQLの実行
- SQL文のファイルへの保存および追加
- 保存されたファイルの実行
- ファイルからバッファへのコマンドのロードおよび編集

ORACLE

Copyright © 2007, Oracle. All rights reserved.

## SQL\*Plus

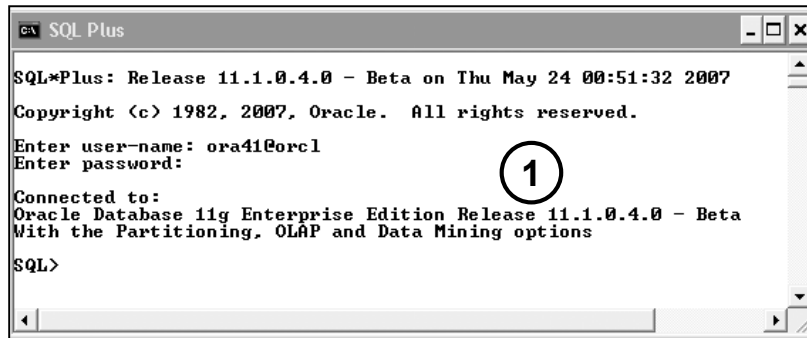
SQL\*Plus環境では、次のことを実行できます。

- SQL文を実行し、データベースに対してデータを検索、変更、追加、削除する
- 問合せ結果の書式設定、計算、格納を行い、レポート形式で出力する
- スクリプト・ファイルを作成し、後から繰り返し使用できるようにSQL文を格納する

SQL\*Plusコマンドは、次の主なカテゴリに分けることができます。

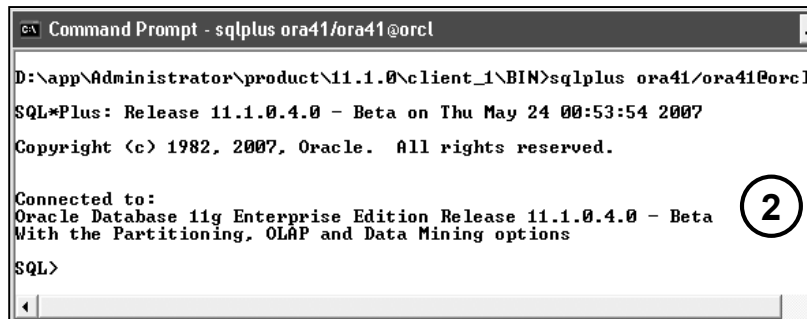
カテゴリ	目的
環境	セッションのSQL文の一般的な動作の設定。
書式設定	問合せ結果の書式設定。
ファイル操作	スクリプト・ファイルの保存、ロード、実行。
実行	SQLバッファからOracleサーバーへのSQL文の送信。
編集	バッファ内のSQL文の変更。
相互作用	変数の作成およびSQL文への引渡し、変数値の出力、メッセージの画面への出力。
その他	データベースへの接続、SQL*Plus環境の操作、列定義の表示。

## SQL\*Plusへのログイン



```
c:\ SQL Plus
SQL*Plus: Release 11.1.0.4.0 - Beta on Thu May 24 00:51:32 2007
Copyright (c) 1982, 2007, Oracle. All rights reserved.
Enter user-name: ora41@orcl
Enter password:
Connected to:
Oracle Database 11g Enterprise Edition Release 11.1.0.4.0 - Beta
With the Partitioning, OLAP and Data Mining options
SQL>
```

```
sqlplus [username[/password[@database]]]
```



```
c:\ Command Prompt - sqlplus ora41/ora41@orcl
D:\app\Administrator\product\11.1.0\client_1\BIN>sqlplus ora41/ora41@orcl
SQL*Plus: Release 11.1.0.4.0 - Beta on Thu May 24 00:53:54 2007
Copyright (c) 1982, 2007, Oracle. All rights reserved.
Connected to:
Oracle Database 11g Enterprise Edition Release 11.1.0.4.0 - Beta
With the Partitioning, OLAP and Data Mining options
SQL>
```

ORACLE

Copyright © 2007, Oracle. All rights reserved.

### SQL\*Plusへのログイン

SQL\*Plusを呼び出す方法は、オペレーティング・システムのタイプ、または実行しているWindows環境によって異なります。

Windows環境からログインするには、次の手順を実行します。

1. 「スタート」→「プログラム」→「Oracle」→「Application Development」→「SQL\*Plus」を選択します。
2. ユーザー名、パスワードおよびデータベース名を入力します。

コマンドライン環境からログインするには、次の手順を実行します。

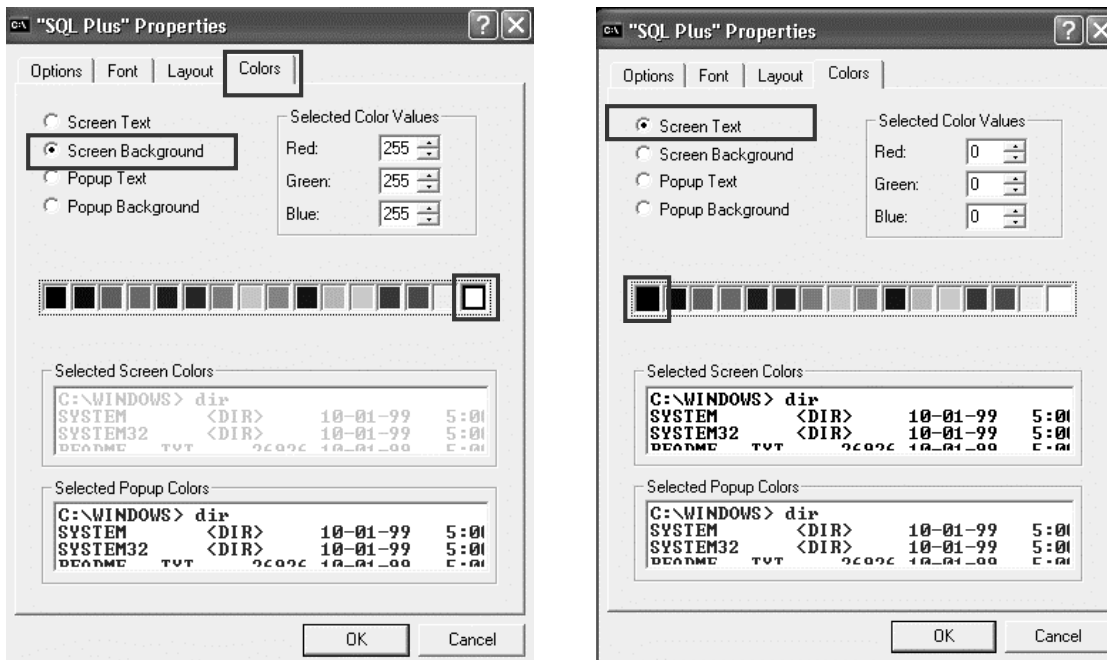
1. マシンにログオンします。
2. sqlplusコマンドを、スライドに示すように入力します。

構文の内容:

<i>username</i>	データベースのユーザー名
<i>password</i>	データベースのパスワード(ここで入力するパスワードは表示されます)
<i>@database</i>	データベースの接続文字列

**注意:** パスワードの整合性を保つため、オペレーティング・システム・プロンプトにはパスワードを入力しないでください。代わりに、ユーザー名のみを入力します。パスワードはパスワード・プロンプトで入力します。

## SQL\*Plus環境の設定の変更



Copyright © 2007, Oracle. All rights reserved.

### SQL\*Plus環境の設定の変更

「SQL\*Plus Properties」ダイアログ・ボックスを使用して、SQL\*Plus環境の表示を任意に変更できます。

SQL\*Plusウィンドウでタイトル・バーを右クリックし、表示されるショートカット・メニューで、「Properties」を選択します。その後、「SQL\*Plus Properties」ダイアログ・ボックスの「Colors」タブを使用して、「Screen Background」と「Screen Text」を設定できます。

## 表構造の表示

表の構造を表示するには、次のようにSQL\*Plus DESCRIBE コマンドを使用します。

```
DESC[RIBE] tablename
```

ORACLE

Copyright © 2007, Oracle. All rights reserved.

### 表構造の表示

SQL\*Plusでは、DESCRIBEコマンドを使用して表の構造を表示できます。コマンドの実行結果には、列名とデータ型が表示され、列にデータが含まれている必要があるかどうかとも示されます。

構文の内容:

*tablename* ユーザーがアクセスできる既存の表、ビューまたはシノニムの名前

DEPARTMENTS表の情報を表示するには、次のコマンドを使用します。

```
SQL> DESCRIBE DEPARTMENTS
```

Name	Null?	Type
-----	-----	-----
DEPARTMENT_ID	NOT NULL	NUMBER(4)
DEPARTMENT_NAME	NOT NULL	VARCHAR2(30)
MANAGER_ID		NUMBER(6)
LOCATION_ID		NUMBER(4)

## 表構造の表示

```
DESCRIBE departments
```

Name	Null?	Type
DEPARTMENT_ID	NOT NULL	NUMBER(4)
DEPARTMENT_NAME	NOT NULL	VARCHAR2(30)
MANAGER_ID		NUMBER(6)
LOCATION_ID		NUMBER(4)

ORACLE

Copyright © 2007, Oracle. All rights reserved.

### 表構造の表示(続き)

スライドの例では、DEPARTMENTS表の構造に関する情報が表示されています。表示結果には次の内容が含まれます。

Null?: 列にデータが含まれている必要があるかどうかを示します (NOT NULLは列にデータが含まれている必要があることを示します)。

Type: 列のデータ型を表示します。

## SQL\*Plus編集コマンド

- A[PPEND] *text*
- C[HANGE] / *old* / *new*
- C[HANGE] / *text* /
- CL[EAR] BUFF[ER]
- DEL
- DEL *n*
- DEL *m n*

ORACLE

Copyright © 2007, Oracle. All rights reserved.

### SQL\*Plus編集コマンド

SQL\*Plusコマンドは、1回に1つの行で入力されます。SQLバッファには格納されません。

コマンド	説明
A[PPEND] <i>text</i>	カレント行の末尾にテキストを追加します。
C[HANGE] / <i>old</i> / <i>new</i>	カレント行の <i>old</i> のテキストを <i>new</i> に変更します。
C[HANGE] / <i>text</i> /	カレント行から <i>text</i> を削除します。
CL[EAR] BUFF[ER]	SQLバッファからすべての行を削除します。
DEL	カレント行を削除します。
DEL <i>n</i>	行 <i>n</i> を削除します。
DEL <i>m n</i>	<i>m</i> から <i>n</i> ( <i>n</i> も含む)までの行を削除します。

### ガイドライン

- コマンドを完了する前に[Enter]キーを押すと、SQL\*Plusによってプロンプトに行番号が表示されます。
- いずれかの終了文字(セミコロンまたはスラッシュ)を入力するか、[Enter]キーを2回押して、SQLバッファを終了します。その後、SQLプロンプトが表示されます。

## SQL\*Plus編集コマンド

- I[NPUT]
- I[NPUT] *text*
- L[IST]
- L[IST] *n*
- L[IST] *m n*
- R[UN]
- *n*
- *n text*
- 0 *text*

ORACLE

Copyright © 2007, Oracle. All rights reserved.

### SQL\*Plus編集コマンド(続き)

コマンド	説明
I[NPUT]	不確定の数の行を挿入します。
I[NPUT] <i>text</i>	<i>text</i> で構成される行を挿入します。
L[IST]	SQLバッファ内のすべての行をリストします。
L[IST] <i>n</i>	1つの行( <i>n</i> で指定)をリストします。
L[IST] <i>m n</i>	<i>m</i> から <i>n</i> ( <i>n</i> も含む)までの範囲の行をリストします。
R[UN]	バッファ内の現在のSQL文を表示して実行します。
<i>n</i>	カレント行にする行を指定します。
<i>n text</i>	行 <i>n</i> を <i>text</i> に置換します。
0 <i>text</i>	行1の前に行を挿入します。

注意: 各SQLプロンプトでは、SQL\*Plusコマンドを1つだけ入力できます。SQL\*Plusコマンドはバッファに格納されません。SQL\*Plusコマンドを次の行に続けるには、最初の行をハイフン(-)で終了します。

## LIST、nおよびAPPENDの使用

```
LIST
1  SELECT last_name
2* FROM   employees
```

```
1
1* SELECT last_name
```

```
A , job_id
1* SELECT last_name, job_id
```

```
LIST
1  SELECT last_name, job_id
2* FROM   employees
```

ORACLE

Copyright © 2007, Oracle. All rights reserved.

### LIST、nおよびAPPENDの使用

- SQLバッファの内容を表示するには、L[IST]コマンドを使用します。バッファ内の行2の横のアスタリスク(\*)は、行2がカレント行であることを示します。編集した内容はカレント行に適用されます。
- カレント行の番号を変更するには、編集する行の番号(n)を入力します。新しいカレント行が表示されます。
- テキストをカレント行に追加するには、A[PPEND]コマンドを使用します。新規に編集された行が表示されます。LISTコマンドを使用して、バッファの新しい内容を確認します。

**注意:** LISTやAPPENDなどの多くのSQL\*Plusコマンドは、省略して最初の文字だけで表すことができます。LISTはLに、APPENDはAに省略できます。



## CHANGEコマンドの使用

```
LIST  
1* SELECT * from employees
```

```
c/employees/departments  
1* SELECT * from departments
```

```
LIST  
1* SELECT * from departments
```

ORACLE

Copyright © 2007, Oracle. All rights reserved.

### CHANGEコマンドの使用

- L[IST]を使用して、バッファの内容を表示します。
- C[HANGE]コマンドを使用して、SQLバッファ内のカレント行の内容を変更します。この例では、employees表をdepartments表に置換します。新しいカレント行が表示されます。
- L[IST]コマンドを使用して、バッファの新しい内容を確認します。

## SQL\*Plusファイル・コマンド

- **SAVE *filename***
- **GET *filename***
- **START *filename***
- **@ *filename***
- **EDIT *filename***
- **SPOOL *filename***
- **EXIT**

ORACLE

Copyright © 2007, Oracle. All rights reserved.

### SQL\*Plusファイル・コマンド

SQL文はOracleサーバーと通信します。SQL\*Plusコマンドは、環境の制御、問合せ結果の書式設定およびファイルの管理を実行します。次の表で説明されているコマンドを使用できます。

コマンド	説明
SAV[E] <i>filename</i> [.ext] [REP[LACE]APP[END]]	SQLバッファの現在の内容をファイルに保存します。既存のファイルに追加するにはAPPENDを使用し、既存のファイルを上書きするにはREPLACEを使用します。デフォルトの拡張子は.sqlです。
GET <i>filename</i> [.ext]	前に保存したファイルの内容をSQLバッファに書き込みます。ファイル名のデフォルトの拡張子は.sqlです。
STA[RT] <i>filename</i> [.ext]	前に保存したコマンド・ファイルを実行します。
@ <i>filename</i>	前に保存したコマンド・ファイルを実行します (STARTと同じ)。
ED[IT]	エディタを起動して、バッファの内容をafiedt.bufという名前のファイルに保存します。
ED[IT] [ <i>filename</i> [.ext]]	エディタを起動して、保存されたファイルの内容を編集します。
SPO[OL] [ <i>filename</i> [.ext]] OFF OUT]	問合せ結果をファイルに格納します。OFFを指定すると、スプール・ファイルが閉じられます。OUTを指定すると、スプール・ファイルが閉じられて、ファイルの結果がプリンタに送信されます。
EXIT	SQL*Plusを終了します。

## SAVE、STARTおよびEDITコマンドの使用

LIST

```
1 SELECT last_name, manager_id, department_id
2* FROM employees
```

SAVE my\_query

Created file my\_query

START my\_query

LAST_NAME	MANAGER_ID	DEPARTMENT_ID
King		90
Kochhar	100	90
...		

107 rows selected.

ORACLE

Copyright © 2007, Oracle. All rights reserved.

### SAVE、STARTおよびEDITコマンドの使用

#### SAVE

SAVEコマンドを使用すると、バッファの現在の内容をファイルに格納できます。この方法で、使用頻度の高いスクリプトを後で使用できるように格納することができます。

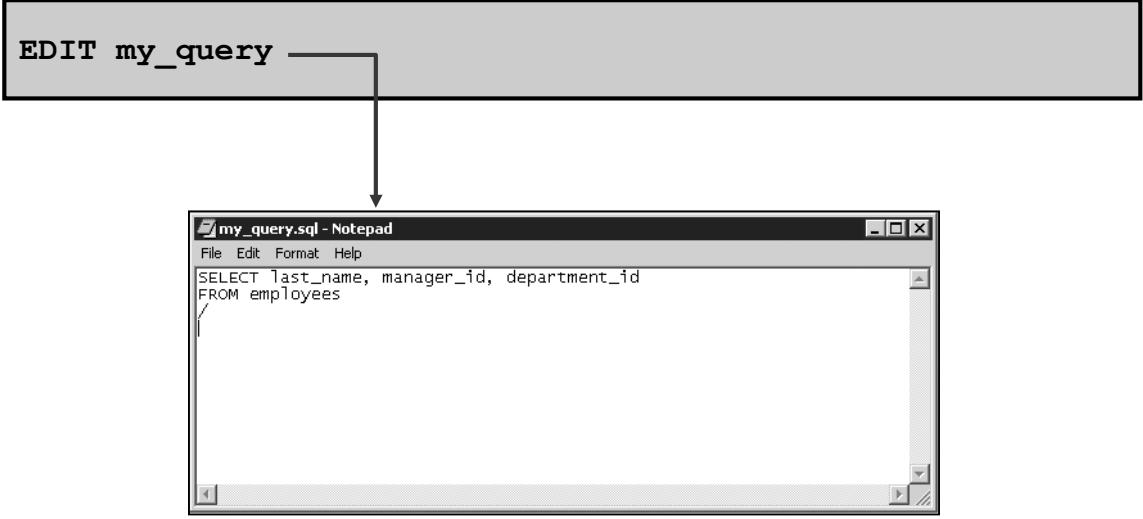
#### START

STARTコマンドを使用すると、SQL\*Plusでスクリプトを実行できます。代わりに、記号@を使用してスクリプトを実行することもできます。

@my\_query

## SAVE、STARTおよびEDITコマンドの使用

EDIT my\_query



ORACLE

Copyright © 2007, Oracle. All rights reserved.

### SAVE、STARTおよびEDITコマンドの使用(続き)

#### EDIT

EDITコマンドを使用すると、既存のスクリプトを編集できます。このコマンドを実行すると、エディタが開かれて、スクリプト・ファイルが表示されます。変更後、エディタを終了してSQL\*Plusコマンドラインに戻ります。

**注意:**「/」は、文の末尾を示すデリミタです。ファイル内で検出されると、SQL\*Plusによって、このデリミタまでの文が実行されます。デリミタは、文の直後に続く新しい行の最初の文字である必要があります。

## SERVEROUTPUTコマンド

- SET SERVEROUT[PUT]コマンドを使用すると、SQL\*Plusでストアド・プロシージャまたはPL/SQLブロックの出力を表示するかどうかを制御できます。
- DBMS\_OUTPUTの行の長さの制限は、255バイトから32767バイトに拡張されました。
- デフォルトのサイズが無制限になりました。
- SERVEROUTPUTが設定されている場合、リソースは事前に割り当てられません。
- パフォーマンスは低下しないため、物理メモリーを節約する必要がない場合はUNLIMITEDを使用します。

```
SET SERVEROUT[PUT] {ON | OFF} [SIZE {n | UNL[IMITED]}]  
[FOR[MAT] {WRA[PPED] | WOR[D_WRAPPED] | TRU[NCATED]}]
```

ORACLE

Copyright © 2007, Oracle. All rights reserved.

### SERVEROUTPUTコマンド

データベース表へのデータの格納またはそれらの表の問合せを行う場合、ほとんどのPL/SQLプログラムは、SQL文を介して入出力を行います。他のすべてのPL/SQL入出力は、他のプログラムと相互作用するAPIを介して実行されます。たとえば、DBMS\_OUTPUTパッケージには、PUT\_LINEなどのプロシージャがあります。PL/SQL外部の結果を表示するには、DBMS\_OUTPUTに渡されたデータの読取りおよび表示を行うSQL\*Plusなどの別のプログラムが必要です。

SQL\*Plusでは、最初にSQL\*PlusコマンドSET SERVEROUTPUT ONを実行しない場合、DBMS\_OUTPUTデータは表示されません。

```
SET SERVEROUTPUT ON
```

#### 注意

- SIZEには、Oracle Databaseサーバー内のバッファに格納できる出力のバイト数を設定します。デフォルトはUNLIMITEDです。nは、2000以上1,000,000以下である必要があります。
- SERVEROUTPUTの詳細は、『Oracle Database PL/SQL言語リファレンス』を参照してください。

## SQL\*Plus SPOOLコマンドの使用

```
SPO[OL] [file_name[.ext] [CRE[ATE] | REP[LACE] |  
APP[END]] | OFF | OUT]
```

オプション	説明
file_name[.ext]	指定されたファイル名に出力をスプールします。
CRE[ATE]	指定された名前で新しいファイルを作成します。
REP[LACE]	既存のファイルの内容を置換します。ファイルが存在しない場合は、REPLACEによってファイルが作成されます。
APP[END]	バッファの内容を、指定したファイルの末尾に追加します。
OFF	スプールを停止します。
OUT	スプールを停止して、コンピュータの標準(デフォルト)プリンタにファイルを送信します。

ORACLE

Copyright © 2007, Oracle. All rights reserved.

### SQL\*Plus SPOOLコマンドの使用

SPOOLコマンドは、問合せ結果をファイルに格納し、必要に応じてファイルをプリンタに送信します。SPOOLコマンドの機能は拡張されています。現在では、既存のファイルに追加したり、既存のファイルを置換したりできます。以前は、SPOOLを使用してファイルの作成(および置換)のみを実行できました。デフォルトはREPLACEです。

スクリプトのコマンドによって生成された出力を、画面に表示せずにスプールするには、SET TERMOUT OFFを使用します。SET TERMOUT OFFは、対話形式で実行されるコマンドからの出力には影響しません。

空白が含まれるファイル名は引用符で囲む必要があります。SPOOL APPENDコマンドを使用して有効なHTMLファイルを作成するには、PROMPTまたは同様のコマンドを使用してHTMLページのヘッダーとフッターを作成する必要があります。SPOOL APPENDコマンドは、HTMLタグを解析しません。9.2またはそれより前のSET SQLPLUSCOMPAT[IBILITY]では、CREATE、APPENDおよびSAVEパラメータは無効です。

## AUTOTRACEコマンドの使用

- SELECT、INSERT、UPDATE、DELETEなどのSQL DML文が正常に実行された場合にレポートを表示します。
- レポートには、実行統計および問合せ実行パスを含めることができます。

```
SET AUTOT[RACE] {ON | OFF | TRACE[ONLY]} [EXP[LAIN]]  
[STAT[ISTICS]]
```

```
SET AUTOTRACE ON  
-- The AUTOTRACE report includes both the optimizer  
-- execution path and the SQL statement execution  
-- statistics
```

ORACLE

Copyright © 2007, Oracle. All rights reserved.

### AUTOTRACEコマンドの使用

EXPLAINを指定すると、EXPLAIN PLANが実行されて問合せ実行パスが表示されます。STATISTICSを指定すると、SQL文統計が表示されます。AUTOTRACEレポートの書式設定は、接続先のサーバーのバージョンおよびサーバーの構成によって異なる場合があります。DBMS\_XPLANパッケージを使用すると、EXPLAIN PLANコマンドの出力を事前定義した複数の書式で簡単に表示できます。

#### 注意

- パッケージとサブプログラムの詳細は、『Oracle Database PL/SQLパッケージ・プロシージャおよびタイプ・リファレンス11gリリース1(11.1)』ガイドを参照してください。
- EXPLAIN PLANの詳細は、『Oracle Database SQL言語リファレンス』を参照してください。
- 実行計画と統計の詳細は、『Oracle Databaseパフォーマンス・チューニング・ガイド11gリリース1(11.1)』を参照してください。

## まとめ

この付録では、SQL\*Plus環境で次のことを実行する方法について学習しました。

- SQL文の実行
- SQL文の編集
- 出力の書式設定
- スクリプト・ファイルとの相互作用

ORACLE

Copyright © 2007, Oracle. All rights reserved.

### まとめ

SQL\*Plusは、SQLコマンドのデータベース・サーバーへの送信およびSQLコマンドの編集と保存を行うための環境です。コマンドはSQLプロンプトまたはスクリプト・ファイルから実行できます。



# SQL Developer GUIを使用した DMLおよびDDL操作の実行

ORACLE

Copyright © 2007, Oracle. All rights reserved.

Oracle Internal & Oracle Academy  
Use Only

## 目的

この付録を終えると、次のことができるようになります。

- SQL Developerのメニュー・オプションを使用したデータ定義言語 (DDL) 操作の実行
- SQL Developerのメニュー・オプションを使用したデータ操作言語 (DML) 操作の実行

ORACLE

Copyright © 2007, Oracle. All rights reserved.

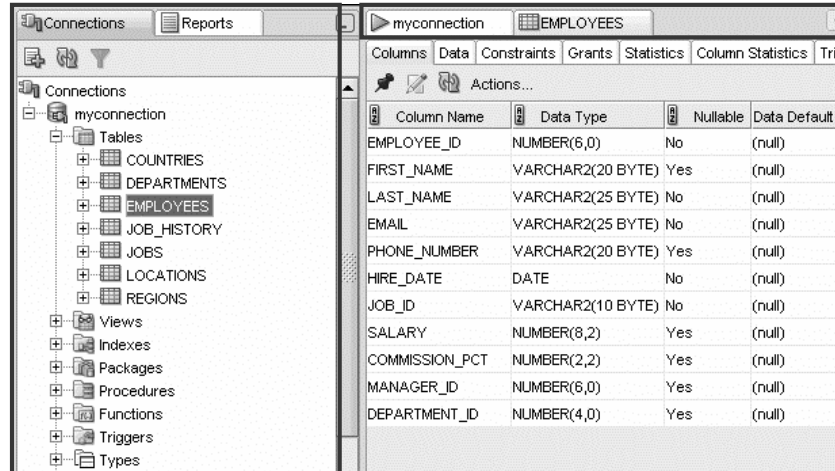
### 目的

この付録では、グラフィカル・ツールであるSQL Developerを紹介します。データベース開発タスク向けのSQL Developer GUIインタフェースを使用する方法について学習します。

この付録は、「はじめに」の章で学習した内容の補足です。すでにデータベース接続を作成して、SQL Developerインタフェース内のオブジェクトを参照できるようになっていることを想定しています。

## データベース・オブジェクトの参照

- データベース接続オブジェクトを作成します。
- 接続ナビゲータを使用すると、次のことを実行できます。
  - データベース・スキーマ内の様々なオブジェクトを参照できる
  - オブジェクトの定義を一目で確認できる



Copyright © 2007, Oracle. All rights reserved.

### データベース・オブジェクトの参照

データベース接続を作成すると、接続ナビゲータを使用して、データベース・スキーマ内の表、ビュー、索引、パッケージ、プロシージャ、トリガー、タイプなどの様々なオブジェクトを参照できます。

SQL Developerでは、ナビゲーションの左ペインでオブジェクトの検索と選択を行います。右ペインには、選択したオブジェクトに関する情報が表示されます。プリファレンスを設定して、SQL Developerの様々な外観をカスタマイズできます。

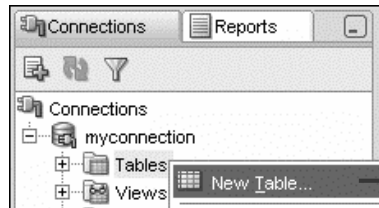
データ・ディクショナリから取り出された情報のタブに分類されているオブジェクトの定義を表示できます。たとえば、ナビゲータで表を選択すると、列、制約、付与、統計、トリガーなどに関する詳細情報が読みやすい形式でタブ・ページに表示されます。

スライドに示されているEMPLOYEES表の定義を表示する場合は、次の手順を実行します。

1. 接続ナビゲータで「Connections」ノードを展開します。
2. 「Tables」を展開します。
3. 「EMPLOYEES」をクリックします。デフォルトでは、「Columns」タブが選択されています。このタブには、表の列の説明が表示されます。「Data」タブを使用すると、表のデータを表示できます。また、新しい行の入力、データの更新、これらの変更のデータベースへのコミットを行うこともできます。

## スキーマ・オブジェクトの作成

- SQL Developerでは、コンテキスト・メニューを使用した任意のスキーマ・オブジェクトの作成がサポートされます。
- 編集ダイアログ・ボックスまたは様々な状況依存のメニューの1つを使用してオブジェクトを編集します。
- DDLを表示して、新しいオブジェクトの作成や既存のスキーマ・オブジェクトの編集などの調整を行います。



「Connections」を展開します。  
「Tables」を右クリックして、  
「New Table」を選択します。

ORACLE

Copyright © 2007, Oracle. All rights reserved.

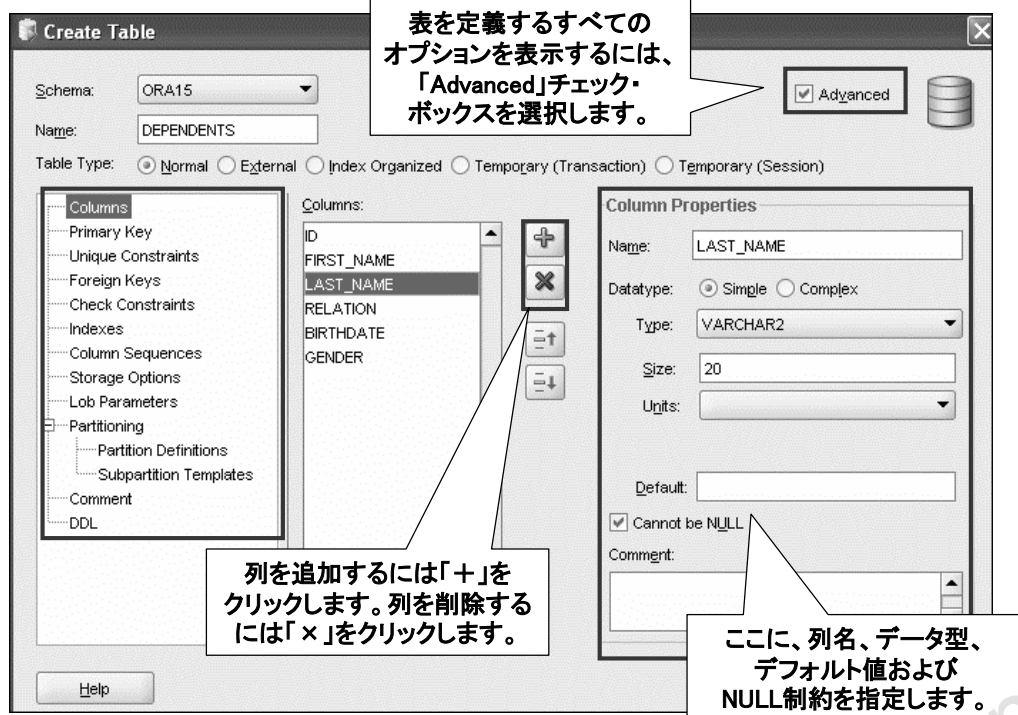
### スキーマ・オブジェクトの作成

SQL Developerでは、SQL Worksheet内のSQL文を実行して任意のスキーマ・オブジェクトを作成できます。または、コンテキスト・メニューを使用してオブジェクトを作成することもできます。作成後は、編集ダイアログ・ボックスまたは様々な状況依存メニューの1つを使用して、オブジェクトを編集できます。

新しいオブジェクトを作成した場合または既存のオブジェクトの編集を行う際、それらの調整に適したデータ定義言語 (DDL) を使用して確認することができます。スキーマ内の1つ以上のオブジェクトに対する完全なDDLを作成する場合は、「Export DDL」オプションを使用できます。

スライドでは、コンテキスト・メニューを使用して表を作成する方法を示しています。新しい表を作成するためのダイアログ・ボックスを開くには、「Tables」を右クリックして、「New Table」を選択します。データベース・オブジェクトを作成および編集するためのダイアログ・ボックスに表示される様々なタブに、そのタイプのオブジェクトに対するプロパティの論理グルーピングが反映されます。

## 新しい表の作成: 例



Copyright © 2007, Oracle. All rights reserved.

### 新しい表の作成: 例

「Create Table」ダイアログ・ボックスでは、「Advanced」チェック・ボックスを選択しない場合、列と使用頻度の高い一部の機能を指定して、表をすばやく作成できます。

「Advanced」チェック・ボックスを選択すると、「Create Table」ダイアログ・ボックスに複数のオプションが表示されて、表の作成時に拡張機能セットを指定できるようになります。

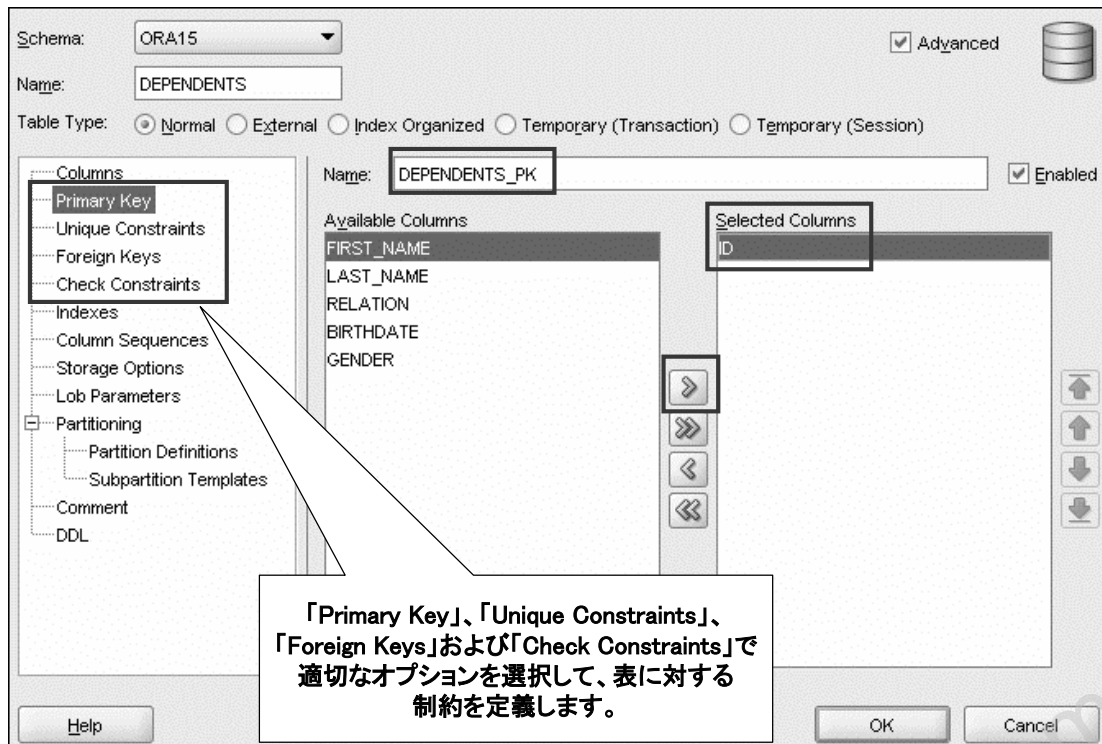
スライド内の例は、「Advanced」チェック・ボックスを選択してDEPENDENTS表を作成する方法を示しています。

新しい表を作成するには、次の手順を実行します。

1. 接続ナビゲータで、「Tables」を右クリックします。
2. 「Create TABLE」を選択します。
3. 「Create Table」ダイアログ・ボックスで、「Advanced」を選択します。
4. 列情報を指定します。「Column Properties」セクションで列名、データ型およびデフォルト値を指定し、「Cannot be Null」チェック・ボックスを選択してNOT NULL列を定義します。列を追加するには「+」をクリックし、列を削除するには「×」をクリックします。スクリーンショットでは、LAST\_NAME列は、VARCHAR2(20)として定義されており、「Cannot be Null」チェック・ボックスを選択することでNOT NULL列として定義されています。
5. 「OK」をクリックします。

必須ではありませんが、ダイアログ・ボックスの「Primary Key」タブで主キーを指定することをお勧めします。作成した表の編集が必要になる場合があります。表を編集するには、接続ナビゲータで表を右クリックして、「Edit」を選択します。

## 制約の定義



Copyright © 2007, Oracle. All rights reserved.

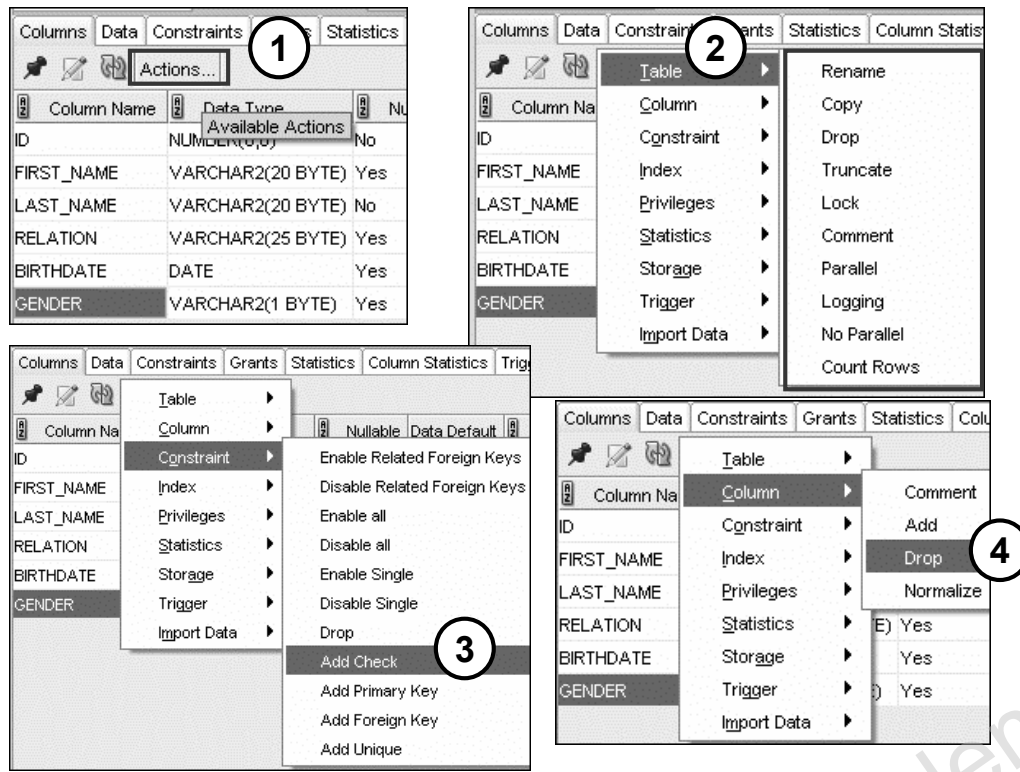
### 制約の定義

同じ「Create Table」ダイアログ・ボックスから、表に対するすべてのタイプの制約を定義できます。スライドでは、「Primary Key」オプションが選択されています。デフォルトの制約名として DEPENDENTS\_PK が割り当てられています。「Available Columns」リストで適切な列を選択し、「>」をクリックして列を「Selected Columns」リストに移動できます。スライドでは、ID が主キーとして指定されています。同様に、「Unique Constraints」オプションをクリックして一意の制約を定義するか、「Foreign Keys」をクリックして外部キーの制約を定義します。「OK」をクリックします。

既存の表に対して制約を定義する場合は、接続ナビゲータで表を右クリックして、「Edit」を選択します。スライドに示されているようなダイアログ・ボックスが表示されます。鉛筆アイコンをクリックして編集することもできます。



## 表の変更



ORACLE

Copyright © 2007, Oracle. All rights reserved.

### 表の変更

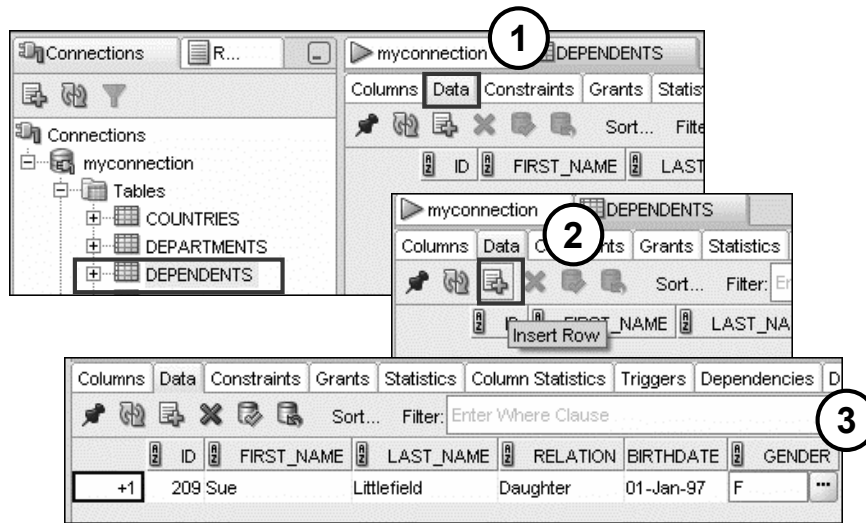
表を作成すると、接続ナビゲータの「Tables」ノードの下に表が追加されます。表の構造を表示したり、表の列の定義を変更したりすることもできます。接続ナビゲータで変更する表をクリックします。右側には、表に関するすべての詳細を示す一連のタブが表示されます。「Columns」タブには表の構造が表示され、「Data」タブには表のデータが表示されます。

「Columns」タブを選択した状態で、次の手順を実行します。

1. 「Actions...」をクリックします。サブメニューが表示されます。
2. 「Table」を選択して、一連のメニュー・オプションを表示します。表の名前を変更する場合は、「Rename」を選択します。または、表からすべての行を削除する場合は、「Truncate」を選択します。表を削除するには、「Drop」を選択します。
3. 「M」および「F」のみを値として受け入れるようにGENDER列に対してチェック制約を追加する場合は、GENDER列が選択されていることを確認してから、「Constraint」を選択して「Add Check」をクリックします。
4. 列を削除する場合は、「Actions」サブメニューから「Column」を選択して、「Drop」をクリックします。

**注意:** 接続ナビゲータで表を右クリックした場合も、同じメニュー・オプションが表示されます。

## 表へのデータの追加



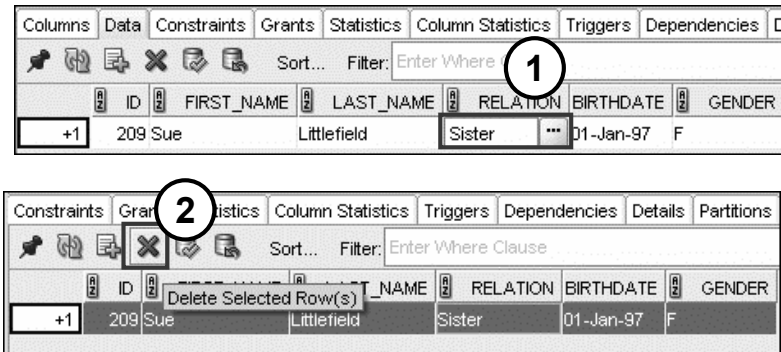
### 表へのデータの追加

表に行を追加するには、次の手順を実行します。

1. 接続ナビゲータで、データを追加する表を選択します。「Data」タブをクリックします。
2. 「Insert Row」アイコンをクリックします。「Insert Row」によって、選択した行の後に空の行が追加され、新しいデータを入力できます。
3. レコードのフィールド値を入力します。別の行を追加するには、「Insert Row」アイコンをもう一度クリックします。



## 行の更新と削除



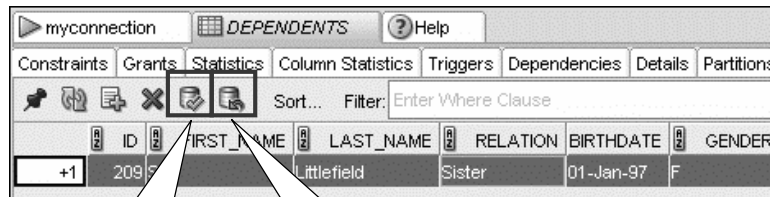
ORACLE

Copyright © 2007, Oracle. All rights reserved.

### 行の更新と削除

1. 表内のデータを更新するには、「Data」タブ内のデータ値のグリッドで直接変更を行います。グリッドにセルを入力すると、様々なデータ型のデータを直接編集できます。また、すべてのデータ型において、省略記号(...)ボタンをクリックしてデータを編集できます。
2. 「Delete Selected Row(s)」をクリックして、削除対象の行にマークを付けます。変更をコミットするまで、実際の削除は行われません。

## コミットとロールバック



トランザクションを保存するには、「Commit Changes」アイコンをクリックします。

トランザクションを元に戻すには、「Rollback Changes」アイコンをクリックします。

ORACLE

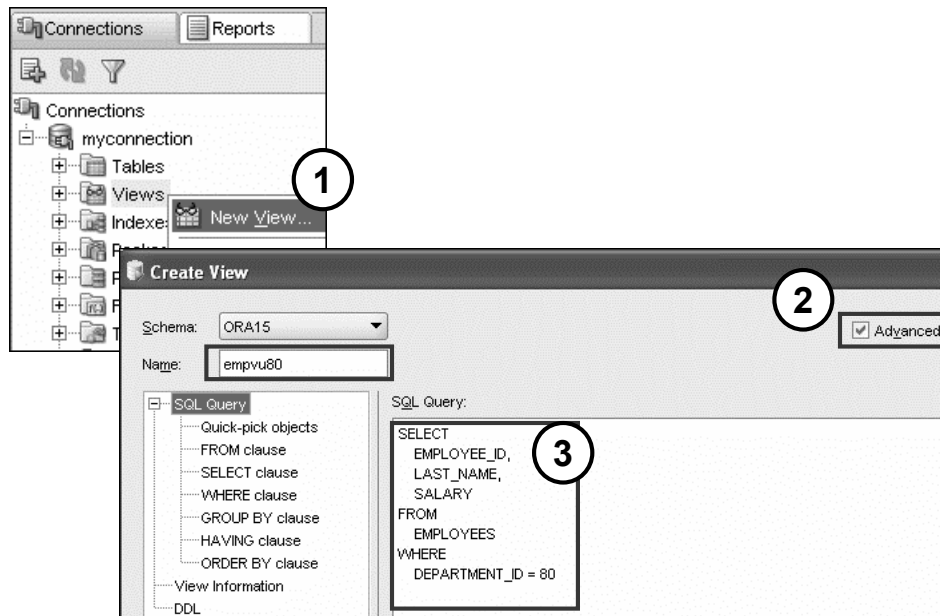
Copyright © 2007, Oracle. All rights reserved.

### コミットとロールバック

新しい行を挿入した場合、またはいずれかの行を更新した場合は、変更を保存するか、元に戻すことができます。

1. 変更を保存するには、「Commit Changes」アイコンをクリックします。「Commit Changes」によって現在のトランザクションが終了し、トランザクションで実行したすべての変更が確定されます。
2. 変更を元に戻すには、「Rollback Changes」アイコンをクリックします。「Rollback Changes」によって、現在のトランザクションで実行された作業が元に戻されます。

# ビューの作成



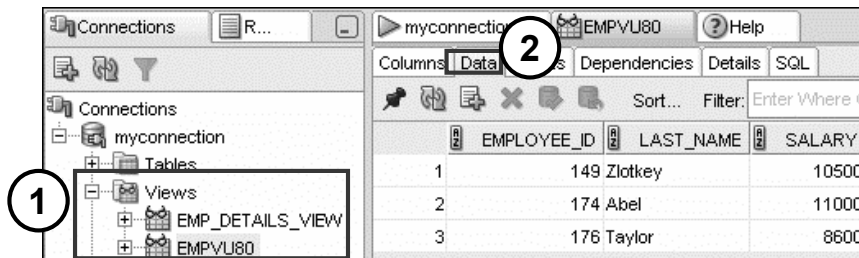
Copyright © 2007, Oracle. All rights reserved.

## ビューの作成

ビューは、基礎となる1つ以上の表からデータを選択する仮想表（一部のデータベース製品での問合せに類似）です。ビューを作成するには、次の手順を実行します。

1. 接続ナビゲータで「Views」を右クリックして、「New View」を選択します。
2. 「Create View」ダイアログ・ボックスで、「Advanced」チェック・ボックスを選択します。このオプションを選択すると、ビューを作成する拡張機能セットを提供するペインがダイアログ・ボックスに表示されるようになります。
3. 「SQL Query」ボックスで、SQL問合せを直接入力できます。または、「SQL Query」ノードを展開して、そのオプションをそれぞれ使用して、段階的にビューを定義できます。「OK」をクリックします。接続ナビゲータ内の「Views」ノードにビューが追加されます。

## ビューのデータの取得



ORACLE

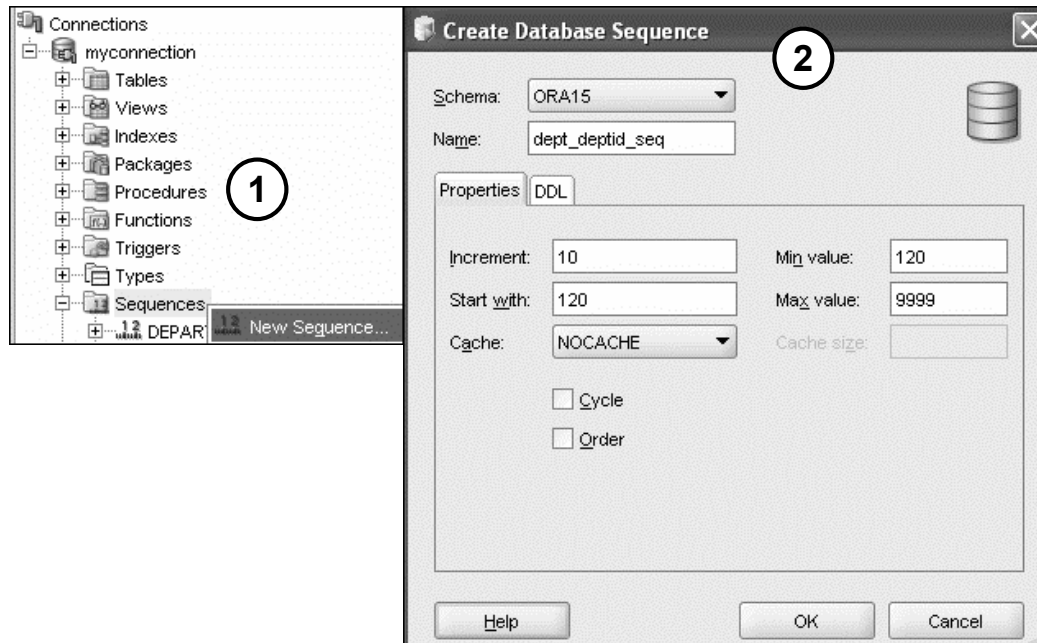
Copyright © 2007, Oracle. All rights reserved.

### ビューのデータの取得

ビューのデータを取得するには、次の手順を実行します。

1. 接続ナビゲータで、「Views」を展開します。データを表示するビューを選択します。
2. 「Data」タブをクリックして、そのデータを表示します。

## 順序の作成



Copyright © 2007, Oracle. All rights reserved.

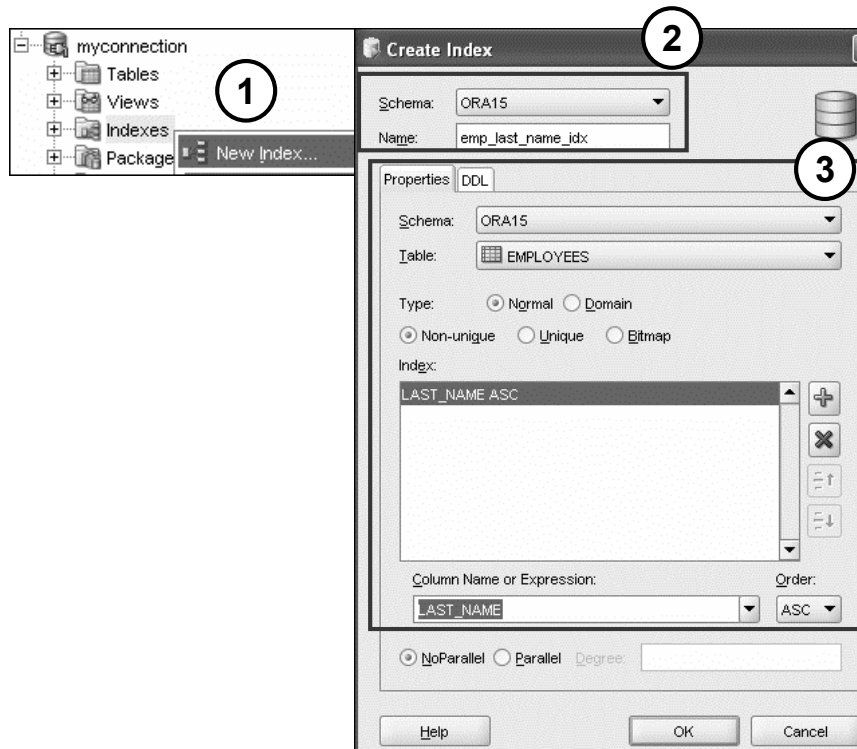
### 順序の作成

順序は一意的な整数を生成するために使用されます。順序を使用すると、主キー値を自動的に生成できます。順序を作成するには、次の手順を実行します。

1. 接続ナビゲータで、「Sequences」を展開します。「New Sequence」を選択します。
2. 「Create Database Sequence」ダイアログ・ボックスで、スキーマ名と順序名を指定します。「Properties」タブ・ページで、増分値、最小値、開始値、最大値などを指定します。この順序のコードの確認および変更を行うには、「DDL」タブをクリックします。「OK」をクリックします。

**注意:** 順序の編集、削除または変更を行うには、順序を右クリックすると表示されるメニュー・オプションを使用します。

## 索引の作成



Copyright © 2007, Oracle. All rights reserved.

### 索引の作成

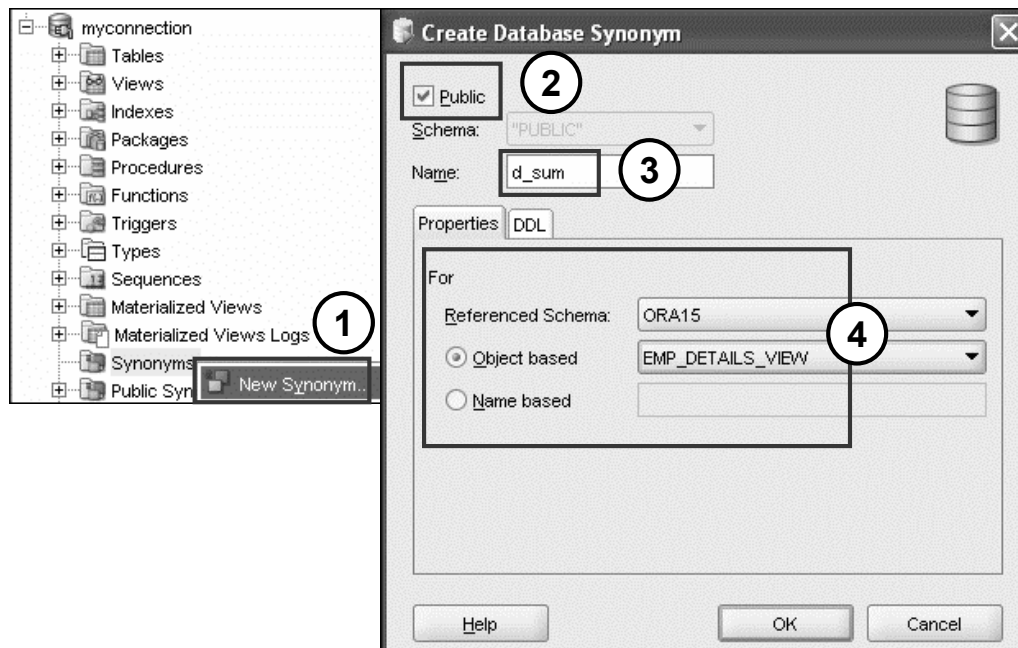
索引は、表またはクラスタの索引付きの列に表示される各値のエントリが含まれるデータベース・オブジェクトです。索引を使用すると、行にすばやく直接アクセスできます。索引は主キー列に自動的に作成されますが、索引付けを効果的に使用するには、他の列に索引を作成する必要があります。

索引を作成するには、次の手順を実行します。

1. 接続ナビゲータで「Indexes」を右クリックします。「New Index」を選択します。
2. 「Create Index」ダイアログ・ボックスで、索引を所有するスキーマを選択します。索引の名前を指定します。
3. 「Properties」タブ・ページで、索引付けする表を所有するスキーマを指定します。索引に関連付ける表を選択します。索引のタイプを選択します。索引の式のリスト(表の列または列の式)を索引に追加します。索引の式を追加するには、「Add Column Expression」(+ )アイコンをクリックします。これによって列名が「Index」と「Column Name or Expression」に追加されます。列名は編集することができます。また、索引の順序を指定することもできます。

**注意:** 索引を右クリックしてから、メニュー・オプションを使用して、索引の編集、削除、再構築および名前の変更を行います。

## シノニムの作成



Copyright © 2007, Oracle. All rights reserved.

### シノニムの作成

シノニムは、表、ビュー、順序または他のシノニムの別名を提供します。接続ナビゲータには、指定した接続に関連付けられているユーザーが所有するすべてのシノニム（パブリックおよびプライベート）に対する「Synonyms」ノードと、接続に関連付けられているデータベース上のすべてのパブリック・シノニムに対する「Public Synonyms」ノードが存在します。

シノニムを作成するには、次の手順を実行します。

1. 接続ナビゲータで「Synonyms」を右クリックします。「New Synonym」を選択します。
2. 「Create Database Synonym」ダイアログ・ボックスで、すべてのユーザーがアクセスできるシノニムを作成する場合は、「Public」チェック・ボックスを選択します。プライベート・シノニムには、そのスキーマ内でのみアクセスできます。
3. シノニムの名前を入力します。
4. 「Properties」タブ・ページで、このシノニムを定義するオブジェクトが含まれているスキーマを選択します。オブジェクトの名前を直接入力するか（「Name based」を選択した場合）、または「Object based」を選択して、スキーマ内のすべてのオブジェクトのドロップダウン・リストを表示できます。リストからオブジェクトを選択します。「OK」をクリックします。

**注意:** パブリック・シノニムは、接続ナビゲータ内の別のノードに追加されます。このため、使用するパブリック・シノニムを「Public Synonyms」ノードで検索する必要があります。シノニムを削除するには、単にシノニムを右クリックして、メニューから「Drop」を選択します。

# スニペットの使用法

スニペットは、単なる構文、サンプルなどのコード・フラグメントです。



Copyright © 2007, Oracle. All rights reserved.

## スニペットの使用法

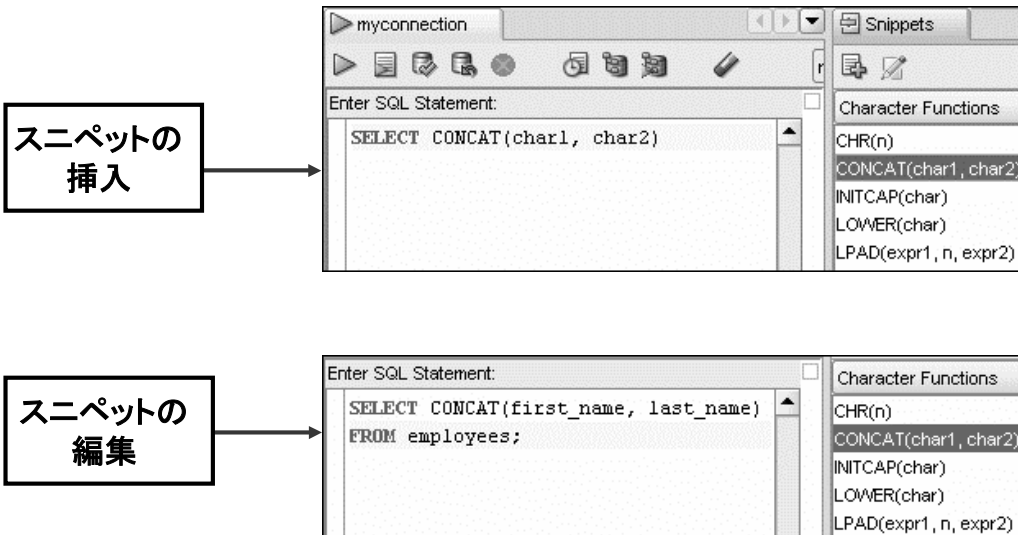
SQL Worksheetを使用する場合、またはPL/SQLファンクションまたはプロシージャを作成したり編集したりする場合は、特定のコード・フラグメントが必要になる場合があります。SQL Developerにはスニペットと呼ばれる機能が用意されています。スニペットは、SQL関数、オプティマイザ・ヒント、その他のPL/SQLプログラミング技術などのコード・フラグメントです。スニペットは、「Editor」ウィンドウにドラッグできます。

スニペットを表示するには、「View」→「Snippets」を選択します。

「Snippets」ウィンドウは右側に表示されます。ドロップダウン・リストを使用して、グループを選択できます。「Snippets」ボタンがウィンドウの右端に表示されるため、「Snippets」ウィンドウが表示されていない場合にこれを表示することができます。



## スニペットの使用法: 例



ORACLE

Copyright © 2007, Oracle. All rights reserved.

### スニペットの使用法: 例

SQL Worksheetのコード、またはPL/SQLファンクションまたはプロシージャのコードにスニペットを挿入するには、スニペットを「Snippets」ウィンドウからコードの目的の場所にドラッグします。これによって、SQL関数が現在のコンテキスト内で有効になるように構文を編集できるようになります。ツールチップでSQL関数の簡単な説明を表示するには、カーソルを関数名に合わせます。

スライドの例は、「Snippets」ウィンドウの「Character Functions」グループからCONCAT(char1, char2)がドラッグされた状態を示しています。ドラッグ後、CONCAT関数の構文が編集されて、文のその他の部分が次のように追加されます。

```
SELECT CONCAT(first_name, last_name)
FROM employees;
```

## 検索エンジンと外部ツール



Copyright © 2007, Oracle. All rights reserved.

### 検索エンジンと外部ツール

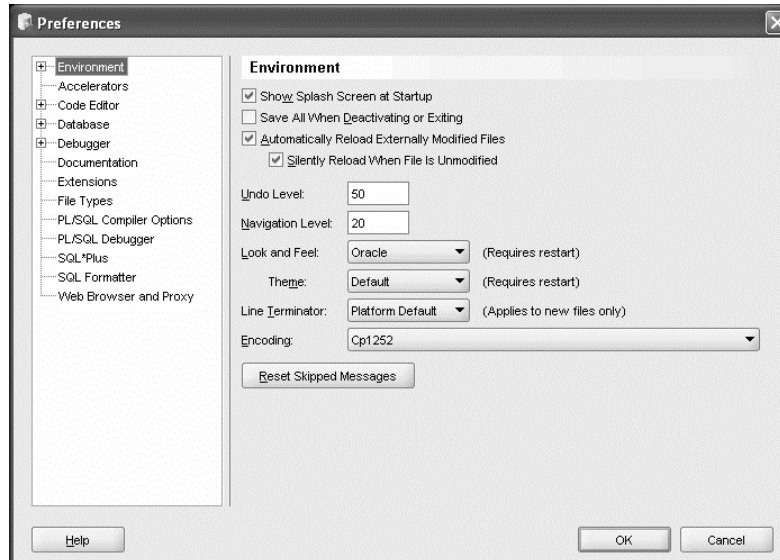
SQL開発者の生産性を向上させるため、SQL Developerに、AskTomやGoogleなどの一般的な検索エンジンおよびディスカッション・フォーラムへのクイック・リンクが追加されました。また、一部の使用頻度の高いツール（メモ帳、Microsoft Word、Dreamweaverなど）へのショートカット・アイコンも用意されています（利用可能な場合）。

既存のリストに外部ツールを追加したり、使用頻度の低いツールへのショートカットを削除したりすることもできます。これを行うには、次の手順を実行します。

1. 「Tools」メニューから、「External Tools」を選択します。
2. 「External Tools」ダイアログ・ボックスで「New」を選択して、新しいツールを追加します。リストから任意のツールを削除するには、「Delete」を選択します。

# プリファレンスの設定

- SQL Developerのインタフェースと環境をカスタマイズします。
- 「Tools」メニューで、「Preferences」を選択します。



ORACLE

Copyright © 2007, Oracle. All rights reserved.

## プリファレンスの設定

要望および必要性に応じてSQL Developerのプリファレンスを変更し、SQL Developerのインタフェースと環境の様々な要素をカスタマイズできます。SQL Developerのプリファレンスを変更するには、「Tools」を選択してから「Preferences」を選択します。

プリファレンスは次のカテゴリにグループ化されています。

- 「Environment」
- 「Accelerators」(キーボード・ショートカット)
- 「Code Editors」
- 「Database」
- 「Debugger」
- 「Documentation」
- 「Extensions」
- 「File Types」
- 「Migration」
- 「PL/SQL Compilers」
- 「PL/SQL Debugger」など

## SQL Developerのチュートリアル

- Oracle SQL Developerを使用して一般的なデータベース開発タスクを実行する方法を学習するには、Oracle SQL Developerのチュートリアルを使用します。
- 次のサイトにアクセスしてください。  
<http://st-curriculum.oracle.com/tutorial/SQLDeveloper/index.htm>

ORACLE

Copyright © 2007, Oracle. All rights reserved.

### SQL Developerのチュートリアル

SQL Developerのチュートリアルは、SQL Developerでの一般的なデータベース開発タスクの実行方法を学習するための広範なリソースです。特定のタスクの段階的な実行方法を視覚的に表示するデモも組み込まれています。

SQLの次に関するトピックまたは節を確認してください。

- データベース・オブジェクトの操作
- データのアクセス
- データの操作

## まとめ

この付録では、SQL Developerを使用して次のことを実行する方法について学習しました。

- SQL Developer GUIインターフェースを使用したデータベース・オブジェクトの参照、作成および編集
- SQL Developer GUIインターフェースを使用した表の行の挿入、更新および削除

ORACLE

Copyright © 2007, Oracle. All rights reserved.

### まとめ

SQL Developerは、データベース開発タスクを簡略化する無償のグラフィカル・ツールです。SQL Developerを使用して、データベース・オブジェクトを参照、作成、編集することができます。また、クイック・メニュー・オプションとグラフィカル・インターフェースを使用して、SQL文で実行できるのと同じデータベース開発タスクを行うことができます。

Oracle Internal & Oracle Academy  
Use Only